MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Sivák Filip** Personal ID number: **393033**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer Vision and Image Processing**

II. Master's thesis details

Master's thesis title in English:

Deep Neural Networks for Classification of Product Images

Master's thesis title in Czech:

Klasifikace obrázků produktů za použití hlubokých neuronových sítí

Guidelines:

The aim of the thesis is to propose, implement, and experimentally evaluate deep neural network architecture [1] for classification of images of various non-food products. The resulting pipeline should include image pre-processing, creation of training, validation, and testing datasets, and thorough experimental evaluation of the proposed architecture(s). Dataset for this task is defined by the Kaggle competition [2]. Implementation should be done in Python with the use of TensorFlow [3]. Integral part of the thesis is analysis of related work. As part of the evaluation, performance comparison with respect to state-of-the-art has to be included. In case the results of the Kaggle competition [2] will be disclosed with sufficient time prior to thesis submission, comparison to best solutions should be included and commented.

Bibliography / sources:

[1] Goodfellow, Ian, et al. "Deep Learning", MIT Press, 2016.
[2] Cdiscount´s Image Classification Challenge, https://www.kaggle.com/c/cdiscount-image-classification-challenge, 2017.
[3] Abadi, Martín, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems, 2015." Software available from TensorFlow. org.
[4] He, Kaiming, et al. "Mask R-CNN" arXiv preprint arXiv:1703.06870 (2017).
[5] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

Name and workplace of master's thesis supervisor:

Ing. Michal Reinštein, Ph.D., Vision for Robotics and Autonomous Systems, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **27.09.2017** Deadline for master's thesis submission: **09.01.2018**

Assignment valid until: **17.02.2019**

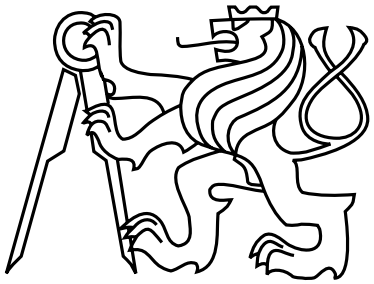
Ing. Michal Reinštein, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics



Master's Thesis

Deep Neural Networks for Classification of Product Images

Bc. Filip Sivák

Supervisor: Ing. Michal Reinštein, Ph.D.

Study Programme: Open Informatics, Master

Field of Study: Computer Vision and Image Processing

January 9, 2018

Aknowledgements

I would like to thank my thesis supervisor Ing. Michal Reinštein, Ph.D. for his support and valuable feedback. I would also like to thank CDiscount company for publishing their dataset and to all Kaggle competitors for sharing their approach.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague on January 1, 2018 .....

Abstract

The high numbers of products and categories on today E-commerce sites render validation of the data as labor intensive and expensive task. Therefore, there is a recent push to automate validation of correct placement of product in category. The French E-commerce company CDiscount has launched Kaggle competition, sharing huge dataset of over 7 million products, To solve the very problem. The goal is to classify products containing multiple images into one of 5270 categories. This thesis proposes, implements and experimentally evaluates deep neural network architecture for classification of non-food E-commerce products. To tackle the complexity of the task on available hardware, hierarchical architecture of neural networks that exploits existing category taxonomy is proposed. The hierarchical architecture achieved the Top-1 accuracy of 0.61061. It has been found, that specific networks in hierarchical architecture can be successfully transferred onto similar datasets, by transferring network that learned on books onto different book dataset. The transfered model performed better than the same model pre-trained on ImageNet dataset.
Keywords: deep learning, classification, product images, E-commerce

Abstrakt

Vysoké množství produktů a kategorií dostupných v současných elektronických obchodech způsobují, že validace dat je pracná a drahá. Proto vzniká snaha automatizovat validaci správného umístnění produktu ve své kategorii. Francouzský elektronický obchod CDiscount proto vyhlásil soutěž hostovanou na Kaggle, kde zveřejnil dataset s více než 7 miliony produkty. Cílem je klasifikovat produkty obsahující jeden či více obrázků do jedné z 5270 kategorií. Tato diplomová práce navrhuje, implementuje a experimentálně ověřuje architekturu hluboké neuronové sítě pro kalsifikaci zboží elektronických obchodů s výjimkou potravin. Pro zmírnění výpočetní náročnosti pro použití na dostupném hardwaru je navržena hierarchická architektura neuronových sítí, která využívá existující hierarchickou taxonomii kategorií. Hierarchická architektura dosáhla Top-1 skóre 0.61061. Přetrénováním neuronové sítě klasifikující obaly knih bylo ověřeno, že specifický model, jenž je součástí hierarchické architektury, může být úspěšně přetrénován pro podobnou trénovací množinu. Model předtrénovaný na knihách dosáhl lepšího výsledku, než totožný model naučený na ImageNet datasetu.
Keywords: hluboká neuronová síť, klasifikace, obrázky produktů, elektronický obchod

Contents

List of Figures

Chapter 1

Introduction

A picture is worth a thousand words. With omnipresent camera and screen equipped devices, images can significantly help users to achieve their tasks. Whether the task is navigating a city using a map, recognizing a person they are about to meet using a photo or choosing the right product to buy, images convey valuable information. On the contrary, misleading images can lead to user frustration. Assessing correctness of an image requires its understanding in a similar way that user understands it. Since manual inspection is labor intensive, hence expensive task [1], there is a significant incentive to automate it.

E-commerce is a fast-growing industry with a notable economic importance that deals with many images. Image of product draws users attention and helps to close a sale. Correct shelving of products on E-commerce sites is an essential prerequisite for successful sales, that is getting harder to fulfill with a rapidly growing number of products.

Chapter 2 of this work describes well-established methods of shelving E-commerce products based on product metadata such as product title and explains why these methods are falling short. The chapter continues with a review of the recent research to include the classification of images as a part of product shelving process. Chapter 3 describes current state of the art Deep Convolutional neural networks for image classification. Chapter 4 describes dataset used as a benchmark of a method developed in this work. Chapter 5 describes design of proposed classification pipeline. The implementation description follows in chapter 6. Implemented models were evaluated in chapter 7. Thesis follows with a comparison of results that were achieved by Kaggle competitors in Chapter 8 and concludes with Chapter 9.

1.1 Overview of methodology

This work concerns classification of E-commerce products based only on their images. Deep learning neural networks were chosen as a solution framework since they provide state-of-the-art performance on image classification. Due to the scale of solved problem, as described in chapter 4, 2the work focuses on transfer-learning from networks pre-trained on ImageNet [2] dataset. The ResNet50 [3] was attempted to train using pre-trained layers, however, the learning was too slow on used GTX 1080 Ti GPU. In order to overcome this problem, only subset of layers of ResNet50 [3] were trained 6.5. To further decrease the complexity of dataset, the category taxonomy present in the dataset is exploited to create a hierarchical

architecture of deep convolutional neural networks 5.6. Top-1 accuracy score of both product images separately and products described in 5.2 is used to evaluate performance of models in chapter 7. The score was chosen to be the same as one used in associated Kaggle competition [4]. The score was evaluated on 20% validation split of labeled data that was not used for training. The overall architecture was evaluated by submitting submission file to Kaggle in order to obtain an evaluation of data without publicly known labels.

1.2 Contribution

Although the task is a Kaggle competition, where users openly share their design ideas, the author of this work have worked independently, forming and testing his own ideas, respecting limitation of his own hardware setup. The contribution of this work follows:

- Design 5.5, implementation 6.2 and evaluation 7.3 of single-model deep neural network architecture
- Duplicate images detection pipeline is proposed in 4.2.1. Found duplicates were analyzed.
- Implementation of dataset preparation pipeline that stores dataset as collection of H5 files was described in 6.2.1
- Design 5.6, implementation 6.4 and evaluation 7.8 of hierarchical deep neural network architecture and all it's respective models
- Comparison of results achieved in books category compared with result of a study [5] on publicly available dataset [6]
- Discussion of solutions of other competitors, that can be found in chapter 8

Chapter 2

Related work

Automatically shelving products on E-commerce site is a nontrivial problem. The task is to find correct category into which product should be assigned. The problem can thus be viewed as multiclass supervised learning task, where the categories are target variables, and product attributes such as title and description serve as features.

Categories can be viewed as departments in a supermarket allowing a customer to navigate and roam around a vast number of aisles. Similarly, as adjacent aisles contain related products, similar categories are connected through a common parent in taxonomy hierarchy. Following the principle of least astonishment, both "mouse" and "keyboard" products are often to be found under shared "computer accessories" category. Carefully crafted taxonomy also helps to build robust recommendation system, that can offer similar products from neighboring categories. Large taxonomy exploitation for personalized product recommendation is studied in [7]. Fine-grained taxonomies can be found outside of E-commerce realm, notable instances being International Patent Classification (IPC) schema or Wikipedia.

2.1 Product classification using text metadata

However, a large taxonomy brings negative repercussions to supervised learning in form of a high number of classes and hence the requirement for large-scale learning. For example, as of 2012 eBay had approximately 20,000 categories covering almost all legally tradeable goods. Their classification engine based on Bayesian classifier with smoothing run daily, using 24,000-word features obtained from product titles as described in [8]. Improvement by ignoring existing hierarchy and learning a new one by discovering latent groups is documented in [9]. Firstly, a product is categorized by coarse KNN classifier lastly, SVM classifier performs fine-grained classification. Neither of the two studies has published used dataset.

Previously mentioned papers were using product titles in a manner of text classification. To identify fundamental differences between product title classification and general text classification was a goal of paper [10]. [11] proposes new performance evaluation metric that tailors to vendor's business goal of maximizing revenue. The proposed metric is used in trained multiclass SVM which utilizes textual features from 5 fields: manufacturer name; UNSPSC code; product name; description; and detailed description. The paper uses a dataset of over 1 million products and classifies into 1073 classes.

More recently [12] uses deep categorization network (DeepCN), which is an end-to-end model using multiple recurrent neural networks (RNNs) dedicated to metadata attributes for generating features from text metadata, which allows diverse attributes to be integrated into a common representation of textual word sequences or nominal values. DeepCN is trained on 94 million items with approximately 4,100 leaf categories obtained from Korean E-commerce website. Six essential metadata attributes are used: item name, brand name, high-level category given by sellers, maker, mall id and image signature. An image signature is represented by nominal value characterizing the color and the edge patterns of an image, allowing image signatures to fit into the common representation.

Another large scale categorization classifier named `TopCat` is presented in [13]. `TopCat` is a probabilistic model leveraging three attributes: product description, price and store.

Notable E-commerce challenge is incompatibility of taxonomies of different providers such as Amazon, Google, eBay, Yahoo! and Walmart. [14] attempts to solve the problem by specifying general taxonomy of 319 nodes organized into 6 levels and training classifier that shelves unlabeled product into it. The paper extract features from product titles and trains classifier on 353,809 examples.

2.2 Product classification using text metadata and images

Textual information about product alone might not be enough to correctly shelve product. Following problems ca occur if relying exclusively on text metadata:

- **Non-descriptive product titles**: While some products such as "Samsung LED TV FULL HD 32" contain discriminating tokens, some titles contain only cryptic designation, such as "Vans OLD SKOOL Black/White", which is the title of boots for men. Cryptic titles mean no harm to the user, who can understand that title describes footwear with one glance at a picture. Some categories tend to use obscure titles, such as "PC motherboards". An example is "Gigabyte GA-970A-DS3P FX".

- **Similar product titles**: There are products where single text token can lead to misclassification yet images are radically different, such as seen in figure 2.1. The figure shows two different products with very similar titles "IBM X201 with battery" and "IBM X201 battery". Moreover, particular example differs in word "with", which is considered to be so-called stopword - frequent English word that would be omitted by feature extraction process for its low information gain.

- **Discrepency in vocabulary usage**: Each merchant tends to use his own language. While some merchants will refer to their line of portable computers as "notebooks", some might prefer "laptops". To differentiate product line, merchants can use original language. More information can be found in [15].

- **Spelling errors**: Textual data can be prone to spelling errors that can lead to misclassification. While images are very sensitive to illumination and view-angle changes, product images are taken in stable conditions each with a single dominant object on white background.

(a) IBM X201 with battery (b) IBM X201 battery

Figure 2.1: Products with similar titles but discriminative images

The Multi-modal deep neural network model for product classification was presented in [16]. The three-part architecture consists of text CNN, image CNN and policy network that learns to chose between the two. The large-scale dataset of 1.2 million instances collected from Walmart.com website is used, shelving products into 2890 possible categories. VGG Network [17] is used as an image CNN.

Confusion Driven Probabilistic Fusion (`CDPF`) is a classifier approach proposed in [15]. `CDPF` trains text classifier and then identifies categories in which the text classifier is highly confused. Top confusing pairs of categories are then found to train three-way image classifiers that resolve, whether product belongs into one of the categories in the pair or into a different category altogether. Study works with a rather small dataset of approximately 27,000 instances classified only to 17 categories.

2.3 Product classification using only images

Influence of image color model to product classification is studied in [18]. The study develops SVM classifier with linear and radial basis function (LaRBF) kernels that classify products into one of 100 categories from PI100 dataset [19] collected from MSN shopping. The dataset contains 10,000 products. Histogram of oriented gradients (HoG) features are used.

Another classical approach is described in recent study [20]. The study uses HoG features as the previous one but conducts dimensionality reduction into eigencolor features. An ensemble of artificial neural networks and SVM were trained in PI100 dataset [19], meaning only 100 categories were used.

To the author's best belief, there is no large-scale study on product classification based solely on images. However, since study [5] trains separate text and image CNN's and report it's respective performances while classifying to 2890 categories, it can be used as a great example of large-scale product image classification.

2.4 Tasks related to product image classification

One of most prominent E-commerce businesses Amazon have started as a bookstore. Books are one of the popular items to buy online. Since books are categorized according to their

genre, product category classification of books is equivalent to genre prediction problem. Hence, to solve product classification problem, one must train a classifier that can classify TV, footwear but also judge book by its cover. The latter is studied in [5]. The study uses the deep convolutional neural network (DCNN) AlexNet [21] that was pre-trained on ImageNet dataset [2]. Large dataset containing 137,788 book cover images in 32 classes was created and published[1].

Notable related problem is generating textual descriptions of product images, which is studied in [22]. Prediction of item popularity on E-commerce site Etsy is studied in [23]. Improving search ranking results using image features is studied in [24].

[1] https://github.com/uchidalab/book-dataset

Chapter 3

Neural networks for image classification

3.1 Artificial neural networks

3.1.1 Intuition

Artificial neural network is a supervised machine learning model. That means the model can learn a function from training examples with labels of desired outputs. Neural networks are inspired by brain function and its ability to learn. Similarly, as there are neurons in a brain that might be seen as biological, computational units, a neural network uses neurons. The intuition behind neural networks is, that although single neuron can learn only elementary function, a million neurons combined can achieve difficult tasks such as recognizing different dog breeds [21].

The neurons are combined into groups called layers, that are connected to each other. The connections between neurons of some layers have associated parameters[1] that change the input value by a function called activation function, resulting in a value called activation. The parameters are subjected to change during the learning process. The layers containing parameters are called trainable layers. An example of a trainable layer is a dense layer that is connecting every input neuron with neurons that it contains. An example of a non-trainable layer is pooling layer described in 3.2.2. Dense layer is therefore parametrized by the number of its neurons and type of activation function. The number of its parameters is dependent on the input layer. The output layer has a shape of expected output. If the task is to predict four numbers, the output layer has three neurons. The layers in between input and output layers are called **hidden layers**. No cycles in network architecture are allowed, meaning the network forms directed acyclic graph.

3.1.2 Obtaining a prediction - forward-pass

The input is passed as a first layer and fed into the network. Each layer takes an input, performs activation function, retrieves an activation and passes it into next layer until the

[1] Also known as weights

output layer returns the result. This process is called forward-pass, as data are passing form an input of network to its output.

3.1.3 Learning a function - backpropagation

The training is done iteratively. In each iteration, the network parameters are changed to improve its performance. To train a neural network, the user must present it with an example of input. At the beginning of learning, the network has no clue how to treat input as the parameters are initialized to small random numbers. The forward pass obtains a result that is most likely wrong. In order to correct the network, the measure of error must be defined, called **loss** or **cost**. The loss is computed by evaluating loss function on desired output and prediction of training example. The learning algorithm then must figure out how to change all the network parameters to decrease the loss. Consequently, negative gradient of loss with respect to parameters is calculated by recursively applying chain rule layer by layer towards the input. This process is called **backpropagation**. This backpropagation is repeated for each example and a fraction called **learning rate** of the average of all obtained negative gradients is then added to weights, updating them. This leads to optimization of all trainable network parameters and convergence to local minima, known as **stochastic gradient descent (SGD)**[25] shown in equation 3.1, where x_{i+1} are new learned parameters, x_i are current parameters, α is learning rate, ∇_x is gradient with respect to x_i and $L(x_i)$ is a loss function.

$$x_{i+1} = x_i - \alpha \nabla_{x_i} L(x) \tag{3.1}$$

3.1.4 Optimizers

There are many variations on SGD, which try to solve various optimization problems that SGD runs into while pushing for convergence. The examples of such problems are:

- Bad local minima is a suboptimal solution that traps optimization process from finding a better solution. Following the steepest gradient might be the reason for problems. Using mini-batches helps to mitigate the problem, as the optimizer does not follow the exact steepest gradient.
- Saddlepoint is opposite of local minima, but have zero gradients as well and can hurt the optimization process, as SGD cannot easily determine which direction to take [26].
- Setting bad learning rate might result in repeatedly overshooting the local minima similarly as golf player overshoots a golf hole.
- A badly scaled dataset can result in a narrow valley in the space of loss function. Instead of quickly descending downhill, the optimization descends to the valley and then goes very slowly through the valley towards minima [27].

High-level overview of various SGD based optimizers:

- Momentum [28], which adds a fraction of the previous update to current update. The momentum behaves like a ball rolling from a hill that acquires momentum.

- Adagrad [29] adapts individual learning rates for each parameter separately by accumulating a sum of squares of all previous gradients of that parameter. This can lead to total progress halt.
- Adadelta [30] improves on Adagrad citeduchi2011adaptive by reducing its monotonically decreasing learning rate by using the only window of gradient accumulation.

3.1.5 Using hardware with high parallelization capabilities

The process of back-propagation as described in the previous section is computationally intensive. However, the training can be significantly sped up by using parallel algorithms. Moreover, using specialized hardware for parallel computation such as graphics cards enables for deep learning to be usable on big datasets that are needed for obtaining models solving demanding tasks such as image classification or speech recognition. The recent development of parallel computational frameworks such as Keras [31] and TensorFlow [32] that can leverage computational power of GPU's have lead to a democratization of neural network usage.

3.1.6 Making learning scalable through mini-batches

The graphics cars employed in deep learning have limited memory[2]. As the whole network containing millions of parameters must be stored in the GPU, not much space remains for training examples. This is tackled by not optimizing trough all examples, but by randomly shuffling dataset and picking a smaller number of examples called **mini-batch** that fits into the memory of GPU. The set of all mini-batches from the shuffled dataset is called **epoch**.

3.1.7 Activation functions

The activation function introduces non-linearity into network, allowing it to learn complex non-convex functions. The two activation functions used in this work are softmax [25] and ReLu [33][3]. The softmax activation is used to perform multiclass classification, as it ensures that all the activations in single layer are summing up to 1. The softmax activation function is depicted in equation 3.2, where K is number of activations, z is vector of activations,

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{for j} = 1,...,\text{K} \tag{3.2}$$

The ReLu activation function returns positive part of its argument, leaving all negative inputs to be zero. Other activation functions can be found in [25].

3.1.8 Over-fitting

The goal of training is to teach classifier a general concept. If the task is to recognize cats, the desired behavior of training process is learning the essence of looking like a cat. However, the neural network will try to achieve the objective of minimizing training loss at all cost. That

[2]For example, high-end gaming graphics card often used for deep learning GTX 1080Ti has 11GB of RAM
[3]Linear rectifier unit

usually means that the network starts to cheat by memorizing training examples instead of learning a general concept. This is called **overfitting**[34]. Ideally, the network should have dataset so big, that it is not able to memorize training examples and is forced to generalize towards the learned concept.

3.1.9 Dropout regularization

One of ways of overcoming overfitting is to use an ensemble of many models. This is however very computationally expensive. The best practice is to use **dropout regularization**[34]. The dropout can be implemented in a form of a layer that randomly stops a portion of activations from propagating. The intuition behind this idea is, that training such network is similar to training multiple networks at once.

3.1.10 Loss functions

The use of loss function depends on the task solved and directly affects speed of training convergence. The most basic problems are binary classification, multi-class classification and regression with their respective loss functions: binary cross-entropy in equation 3.3 where n is number of examples, y_i is target label, $f(x_i, \theta)$ is classification function with input x_i and parameters θ.

$$-\sum_{i=1}^{n} y_i log f(x_i, \theta) + (1 - y_i) log(1 - f(x_i, \theta)) \tag{3.3}$$

Cross categorical entropy in 3.4 where n is number of examples, y_i is target label, $f(x_i, \theta)$ is classification function with input x_i and parameters θ.

$$-\sum_{i=1}^{n} y_i log f(x_i, \theta) \tag{3.4}$$

Square loss in 3.5 where n is number of examples, y is target value and $\hat{y}$ is prediction.

$$-\sum_{i=1}^{n} (y - \hat{y})^2 \tag{3.5}$$

3.2 Convolutional neural networks

3.2.1 Convolutional layers

Convolutional layers [35] are neural network layers preserving the spatial structure, which is the primary difference from traditional fully connected neural network layers. Having, for example, $32 \times 32 \times 3$ image, instead of stretching it to the one-dimensional vector of 3072 items, the image is kept in its original 2D structure. By applying the convolutional filter, the input is transformed into a different tensor called activation map that also preserves structural properties. Since activation maps can be convolved again without loss of structural

information, stacked convolutional layers can be used for dimensionality reduction of spatial data into low-dimensional feature-rich vector space where conventional fully connected networks can be applied.

The filters are small matrices of numbers that are multiplied by regions of input. For every pixel in the input layer, the center of the filter is aligned to it and filter is multiplied with the region of input of the same size as the filter. This process is repeated for all pixels except for those not having a sufficiently big neighborhood, resulting in activation map of slightly smaller size. Repetition of applying the filter over the image can be viewed as a filter sliding over the image, hence convolution. A filter always extends the original depth of input. If the input is to be an RGB image, all filters applied to that image will have the depth of 3.

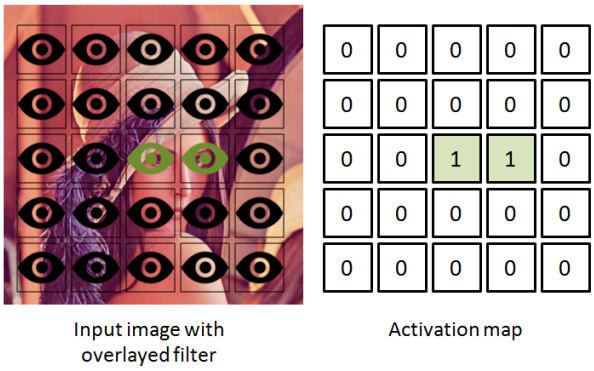


Figure 3.1: Convolution schema

Besides changing filter size, we can adjust the way filter is sliding over an input. Instead of going over every pixel, the filter can be applied over only every other pixel. This parameter is called **stride**[25]. Sliding filter over every pixel means stride is of value 1, and missing every other pixel means stride is of value 2. Naturally, the higher the stride, the smaller resulting activation map gets. It is important to note, that not every input size, filter size, and stride are compatible. To remedy this shortcoming, one can zero pad the input to achieve compatible parameters. This also helps to reduce the rate of area shrinking, as high rate of area shrinking leads to too rapid loss of information. Stride can also be seen as a resolution of filter's view of input volume.

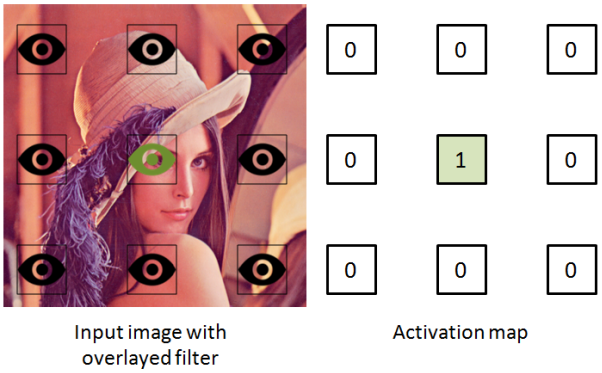


Figure 3.2: Visualisation of stride

3.2.2 Pooling layers

An alternative approach to shrink input volume area is to use a pooling layer [25]. Pooling layer performs aggregations over regions instead of multiplication with filters of trained weights. Usually performed aggregation is maximum of the region, giving the name of the max-pooling layer. The intuition behind effectivity of max-pooling layer in classification task is that it does not matter where in the region have feature been found as long as it has been found. Taking the maximum of a region of activations disregards unimportant parts of that region and reports presence of the feature in the whole region. If averaging were to be used instead of maximum, the fact that feature was not detected in rest of the region would have a negative effect on significant activations. Pooling layer has no trainable parameters. Most common usage of pooling layer is down-sampling; therefore stride is set up so that regions are not overlapping.

A single filter is looking for a particular feature in the input. There are many features to be found in an image. Therefore many filters are needed. Considering the usage of multiple filters, each filter results in its own activation map, together yielding a stack of activation maps called output volume. This way, an input image can be transformed into much deeper volume (keep in mind the difference between the depth of volume meaning a third dimension and depth of neural network as a number of layers). Intuitively, as input is being transformed from input image towards features across the network, the area of input volumes is decreasing with applied stride and/or pooling, while the depth of input volumes can both increase and decrease based on the number of used filters in convolutions. To summarize, single convolutional layer requires four hyperparameters: number of filters, filter size, stride and amount of zero padding.

3.3 Transfer learning

Transfer learning [36] is a way of transferring knowledge of trained network to a different problem. Convolutional neural networks learn function that encodes input image in gradually smaller representations in order to perform successful predictions. The last fully connected layers of the network can be seen as a separate classifier that learns on representation prepared

by previous layers. This compact representation of images can be exploited for other tasks, such as image retrieval [37]. The classifier can be replaced with different fully connected network with different target variables to solve a different task. In this process, features from so-called bottleneck layer that precedes fully connected network called **classification head** are often extracted beforehand for the whole dataset, in order to spare time taken by forward pass.

In the figure 3.3 transfer learning of model trained on dog images can be seen. The model is transferred to recognize cat images. All convolutional layers are intact during training, with retraining only cat classifier with the usage of learned dog features. During the transfer learning, an assumption of the closeness of datasets is important.

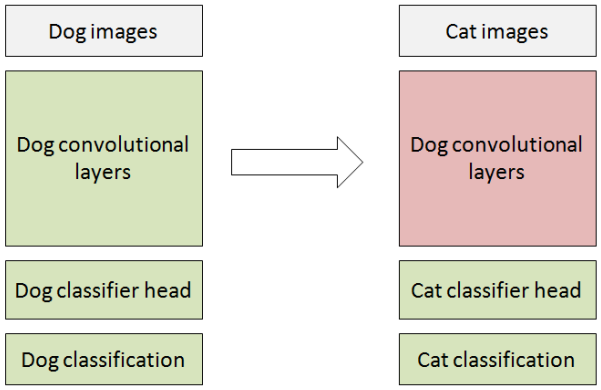


Figure 3.3: Transfer learning schema

3.4 Transfer learning with fine-tuning

The pre-trained network can be also used as a weight initialization for training. The top layers of convolutional networks describe very general features useful in many domains as so are often exempted from training. Such layers are called frozen. Re-training of a pre-trained network is referred to as **fine-tuning**. In the figure 3.4 is the same problem as in the previous section, but this time, some convolutional layers are allowed to change from dog features to cat specific features, allowing to gain higher performance.

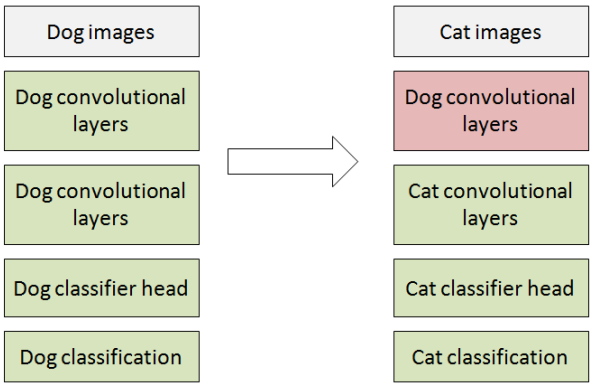


Figure 3.4: Transfer learning with fine-tuning schema

3.5 Deep convolutional neural network architectures

3.5.1 LeNet

LeNet [35] was one of the first instantiations of CNN that was successfully used in practice. Applying 5×5 convolutional filters at stride 1 with 2×2 pooling layers applied at stride 2 between them and finished with two fully connected layers, LeNet was very successfully applied to do hand-written digit recognition. Network architecture can be seen in figure 3.5.

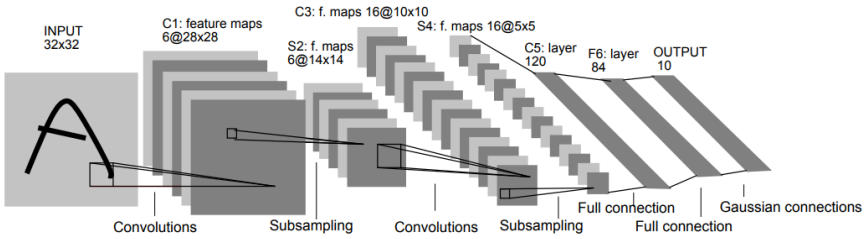


Figure 3.5: LeNet architecture schema. Reused from [35]

3.5.2 AlexNet

AlexNet [21] was first large-scale CNN network that was able to surpass other methods in ILSVRC [38] competition in 2012 by a significant margin. The success of this network sparked a new wave of CNN research that lasts until today. AlexNet is quite similar in architecture to LeNet as we can see in figure 3.6. The significant difference is in its depth and thus overall network complexity and capacity. AlexNet was also the first network to use now often used ReLU activation functions. Becuase GTX 580 GPU's, with only 3 GB of memory, were used, network was split over two GPU's, training half of convolutional feature maps on one device and the other half on the second. Only in last few layers the GPU's pass over their information to do conjoint backpropagation.

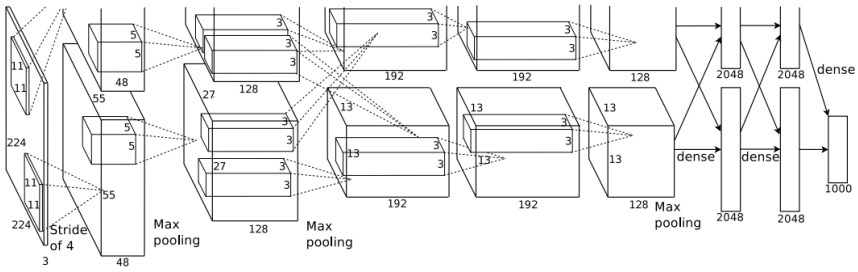


Figure 3.6: AlexNet architecture schema. Adapted from [21]

3.5.3 GoogLeNet and VGG

The notion that deeper is better confirmed two architectures that took on ILSVRC challenge in the year 2014: GoogLeNet [39] from Google and VGG [17] from Oxford. VGG comes with two times the layers than AlexNet had: 16. GoogLeNet is again much deeper network with 22 layers. However, its main insight is in using "inception" modules to reduce the computational difficulty. Even tough GoogLeNet is deeper network than AlexNet; it has twelve times fewer parameters. The inception module is local network topology that can be seen as a network within a network. The input of inception module is fed into multiple different layers such as convolutions of different sizes and pooling. The layers have stride and padding set up so that resulting activation volumes have the same area. The activation volumes are depth-wise concatenated, forming an output of the module.

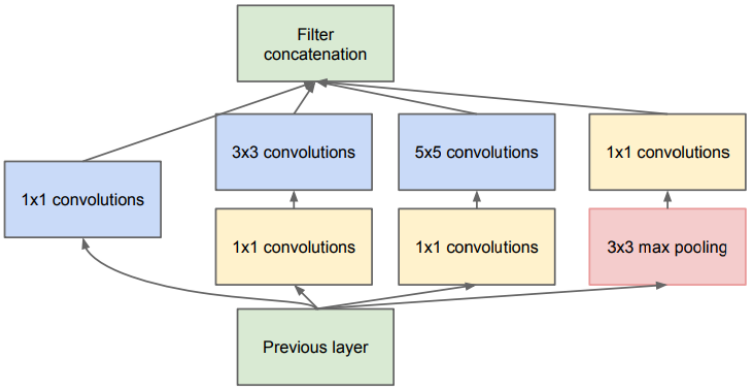


Figure 3.7: GoogLeNet Inception module detail. Adapted from [39]

Because one of the layers inside inception module is pooling and its output is concatenated with other layers, the depth of output would be always higher than the input. Since inception modules are stacked, this would lead to rapid depth increase and high computational difficulty. These problems are overcome by using "bottleneck" layers to reduce the dimensionality of input or output of layers within inception module. The implementation of bottleneck layer is

1×1 convolution with the number of filters corresponding to the desired depth. The input of GoogLeNet is first run through a small conventional network of convolutions and pooling before entering stack of inception modules. There are also two auxiliary outputs branches to inject gradients in earlier levels, which are discarded during inference.

1×1 convolution is meaningful and useful layer used to reduce the dimensionality of an input volume. As expected, because the size of convolution is 1×1, the area of input volume is not changed. On the contrary, the depth changes, since its dependent on the number of used filters. For example, if input volume is of dimensions $56 \times 56 \times 64$ and $1 \times 1 \times 32$ filter is applied, each pixel input volume having 64 different channels is multiplied with 64 weights present in the filter. Because there are 32 such filters, resulting in the volume having the size of $56 \times 56 \times 32$, hence reduced dimensionality. The advantages of using 1×1 convolution to reduce dimensionality are multiple:

- Network learns how to do the reduction the best way on its own using backpropagation
- Layer introduces additional nonlinearities
- It is just a convolutional layer

Although the layer is convolutional, its function is the same as of a fully connected layer.

3.5.4 ResNet

The 2015 ILSVRC challenge won network named ResNet [3], having almost seven times deeper network of 152 layers. The hypothesis of ResNet's authors was, that the deeper the models are, the harder it gets to optimize. The remedy the authors found is an introduction of residual learning. Similarly, as in GoogLeNet, ResNet is composed of stacked modules. Each module consists of two stacked convolutional layers, with the input of module being added to its output as seen in figure 3.8. This means that output of module can be expressed as $H(X) = F(X) + x$, where x is module input, $H(x)$ is module output and $F(x)$ is function learned by the two convolutional layers, called "residual", hence the name Residual network - ResNet. The intuition behind the idea of learning residuals is, that it is easier to optimize learning of residuals which are close to identity functions than learning $H(x)$ directly. Whether the intuition is correct or not, the network architecture works well as demonstrated in ILSVRC challenge. The ResNet is used with a depth of 34, 50, 101 or even 152, with the deeper ResNets using bottleneck layers used in GoogLeNet, named Resnet34, Resnet50 and so on. The work is continued in [40] improving ResNet block design that gives better performance. Another example of improvement can be found in Wide residual network [41] that uses more filters in convolutional layers, hence "wide network". The study was able to show that wide ResNet50 has better performance than original ResNet152, which is deeper. The ResNeXt network [42] introduces parallel pathways to the network modules in a similar fashion as GoogLeNet.

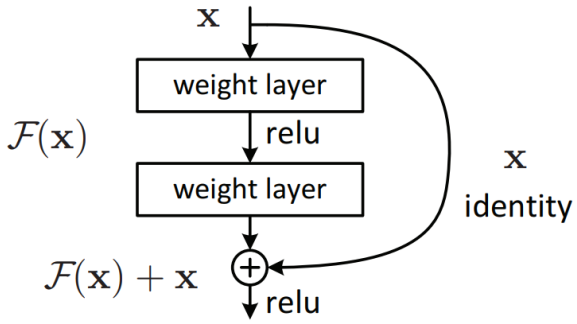


Figure 3.8: ResNet module schema. Adapted from [3]

Other more recent architectures include Stochastic depth network [43] that look very similar to ResNet, but randomly drops a subset of modules bypassing them with identity to reduce depth while training, but using the deeper network at test time. This can be seen as dropout applied to layers across the whole network as opposed to dropout within the single layer as described earlier. The FractalNet [44] works similarly, dropping out pathways in their fractal module containing pathways of various depth, but not using any residuals. The Densely connected convolutional network [45] introduces "dense block" with layers that have multiple outputs connected to inputs of layers deeper in the network, contributing their learned features and reducing their redundancy and mitigating vanishing gradients problem.

3.5.5 InceptionV4

The architecture GoogLeNet was improved in subsequent work [46] where authors introduce batch normalization into the network, calling it InceptionV1. In the following study [47] InceptionV2 and InceptionV3 were proposed. The inceptionV2 introduced convolution factorization, which means they have replaced 7×7 convolutions with 3×3 convolutions having the same effective reception field while introducing more non-linearities and having fewer parameters. InceptionV3 is a variant of InceptionV2 that adds BN-auxiliary, in which additional normalization in the fully connected layer of an auxiliary classifier is used. Finally, study [48] introduces InceptionV4, which is a streamlined more uniform version of previous architecture, where auxiliary classifiers were dropped.

3.5.6 Mask RCNN

The Mask-RCNN contribution of Facebook AI research proposed in study [46] provides a model to perform challenging task of instance segmentation. It is a combination of commonly used approaches for object detection, which is to use Faster-RCNN model [49] for object detection and fully convolutional network [50] to classify instances as parallel heads to their trained feature extractor. The feature extractors of choice were ResNet101 and ResNeXt-101. They also introduce RoIAlign which is an improvement over RoIPool operation without using quantization to avoid information loss. The region is mapped into the feature map and is bilinearly interpolated into fixed-dimensional RoI output. The authors point out, that RoI

pool was not designed to do object segmentation but to do object detection, and so does not produce good masks. Extra convolutional layers on top of RoI output are used to obtain the final accurate mask.

Chapter 4

Dataset

Experiments were conducted on the huge dataset of non-food products that was distributed as a part of Kaggle competition `Cdiscount's Image Classification Challenge` [4]. The data are provided by French E-commerce site Cdiscount. The dataset is composed of almost 9 million products, each having 1 to 4 RGB images of resolution 180x180 pixels and is split to annotated part for training and un-annotated part for contest validation. Each product is labeled by three levels of categories. Top level has 49 different labels, middle level has 483 labels, and finally bottom level has 5270 different labels, which is prediction target of the Kaggle competition. Dataset structure is shown in figure 4.1. An example of hierarchical labels for top level category `DVD  BLUERAY` can be seen in figure 4.2. Dataset statistics can be found in table 4.1. The task is formulated as image classification, and no other data than images are present in the dataset. All names of categories are in the French language. The dataset consists of three files:

- train.bson with size of 58,1 GB
- test.bson with size of 14,5 GB
- category_names.csv containing category taxonomy

Number of training products	7,069,896
Number of training images	12,371,293
Number of test products	1,768,182
Number of test images	3,095,080
Image size	180 x 180 x 3
Number of level one categories	49
Number of level two categories	483
Number of target classes	5270
Image encoding	JPEG
Size	58,1 GB
Format	bson (mongodb)

Table 4.1: Train dataset statistics

Description	Number of products	Number of categories	References
Cdiscount dataset used in this work	**7 million**	**5270**	**[4]**
PI100 dataset from MSN shopping	10,000	100	[19], [18], [20]
Downloaded from Walmart.com	1.2 million	2890	[16]
Bing Shopping	27,000	17	[15]

Table 4.2: Comparison with other datasets

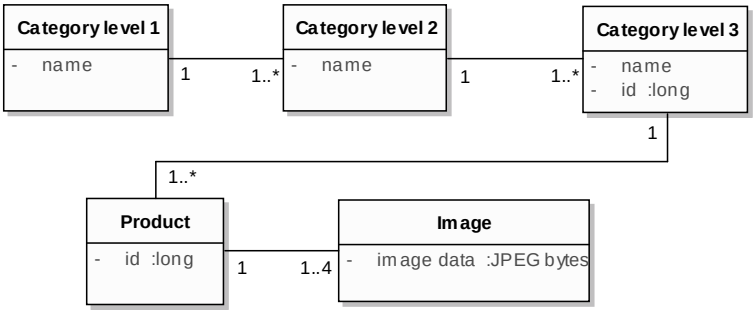


Figure 4.1: Dataset structure

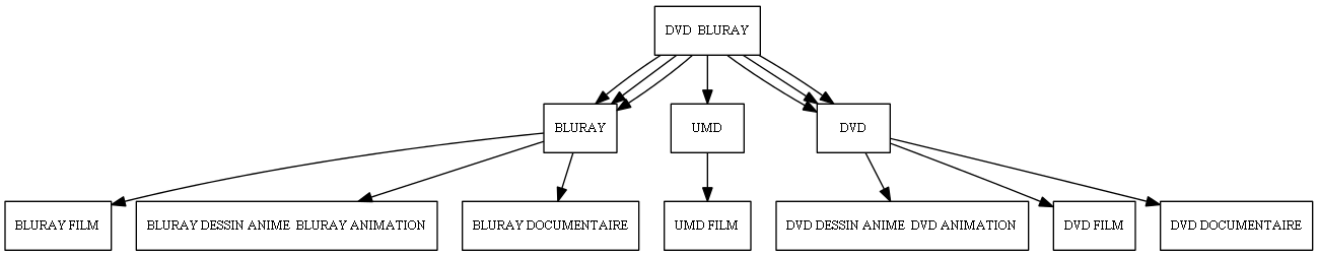


Figure 4.2: An example of hierarchical labels. Most of level 3 categories omitted for brevity.

4.1 Dataset details

Categories in the dataset have unbalanced product counts as can be seen in figure 4.5, where categories are sorted from ones with most products to least. There are 1778 categories with less than 100 products, and 134 categories contain more than half of all products. Table 4.4 shows five largest categories.

Most categories contain products that can have 1 up to 4 images, with most products having only one image as seen in figure 4.3. There are also 34 categories with products containing exactly one image, one category containing exactly 2 images for every product and 3 categories

containing exactly 4 images for all their products. More details can be seen in table 4.3. An example of a product containing 4 images can be found in figure 4.4.

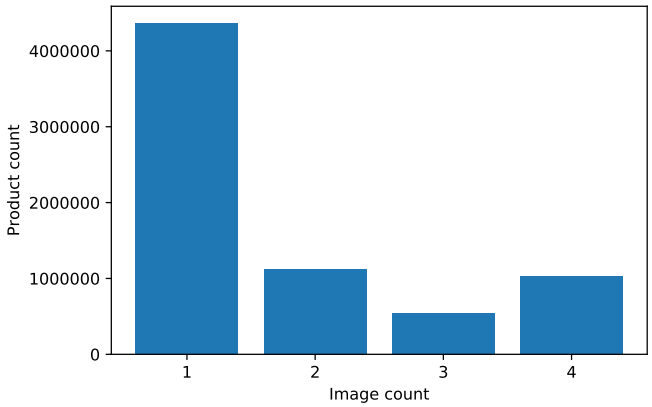


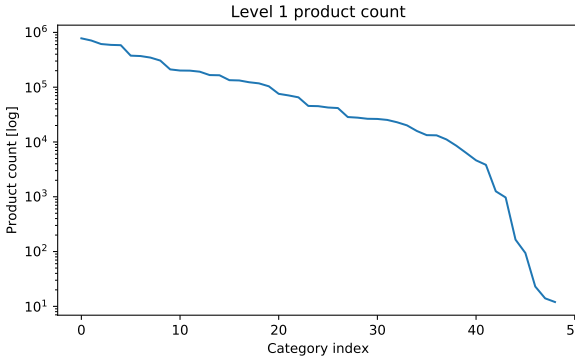
Figure 4.3: Product image count distribution

Figure 4.4: Product with 4 images.

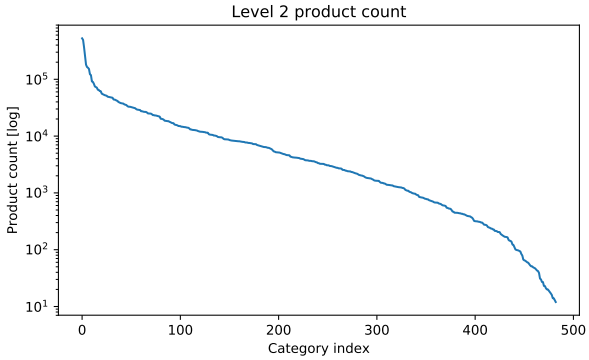
Min image count	Max image count	Number of categories
1	1	34
1	2	170
1	3	169
1	4	4885
2	2	1
2	4	5
3	4	3
4	4	3

Table 4.3: Category image count

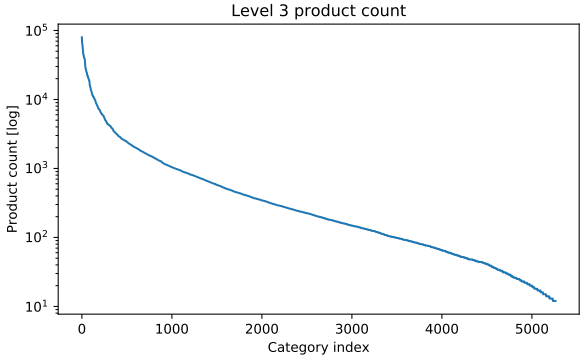
Dataset is stored in `mongodb` [51] database represented by binary `bson` file format. File can be read using `pymongo` [52] package either using linear iterator or by random access reading using seek and precomputed offset table as can be seen in an example notebook availible on Kaggle [53].



(a) Category level 1



(b) Category level 2



(c) Category level 3

Figure 4.5: Logarithmic plots of product counts in different levels of categories

category_level3	category_id	product_count
CD POP ROCK - CD ROCK INDE	1000018296	79640
TONER - RECUPERATEUR DE TONER	1000011423	71116
CARTOUCHE IMPRIMANTE	1000011427	69784
LITTERATURE FRANCAISE	1000014202	65642
AUTRES LIVRES	1000015309	65435

Table 4.4: Largest level 3 categories

4.2 Duplicate images in dataset

The dataset contains 922,822 different duplicated images. That is 5521232 files that are duplicated somewhere in the dataset. An interesting example is a category of smartphone covers, where approximately 16,000 out of 21,649 products share same three out of four images that show instructions how to attach the cover to the phone as can be seen in figure 4.6. These three images are used only in the one category and thus can likely help achieve

good classification results. Other examples of such images can be seen in figure 4.7. Most of these images seem to signal to the user that there are more color variations of the depicted product.

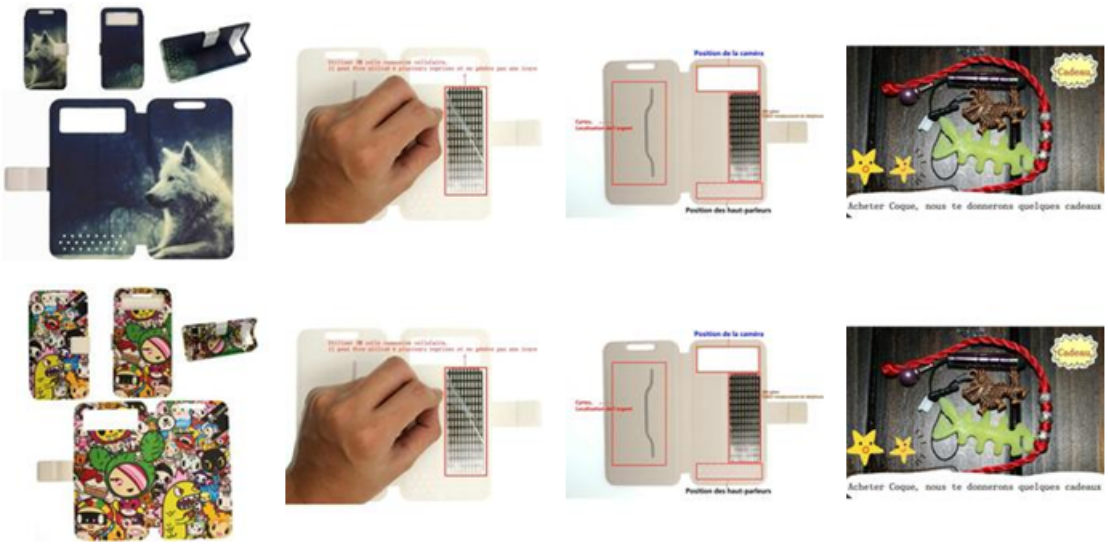


Figure 4.6: Products sharing three same images.

Figure 4.7: Products with duplicates within single category.

On the other side, there is a duplicate image shared among 429 different categories with 15013 duplications that are depicted in figure 4.8. The dataset also contains 1907 completely white images in 260 different categories and 828 images of a question mark in 139 categories. Many other placeholder images can be seen in figure 4.9.

Figure 4.8: Image with caption saying "Image not available" that can be found in 429 different categories

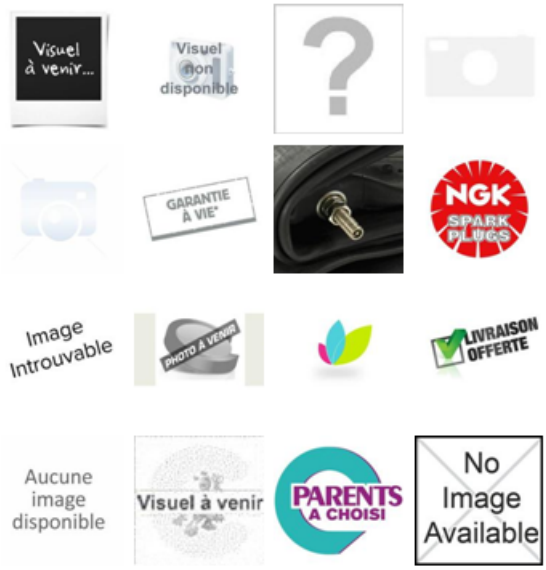


Figure 4.9: Various placeholder images spanning multiple categories.

4.2.1 Finding duplicates

- An image was loaded as an array of bytes
- MD5 hash of array was computed using `hashlib` [54] python package and converted to hex digest
- MD5 hex digest was shortened to 10 letter long string by picking 10 random indexes that are shared for all images. This was done so that resulting representation is small and easy to fit into RAM.

- For each shorten digest, list of product id's of products with the same digest was created, resulting of list of products with at least one duplicate image

4.3 Difficult training examples

A random sample of training example images was drawn and manually inspected for possible problems. Problematic images can be seen in figure 4.10. Description of each image labeled by letters A to F follows:

A) The image represents hair extension, which is a strip of hair that customer can attach to his or her hair to make them appear longer. The customer wears the product to deceive observers into thinking that customer has long hair. The main point of the product is not to be recognized, which works against the goal of this work.

B) The logo of product is an important feature, which explains why someone would like to buy a sticker with such logo, that could potentially confuse a classifier

C) One of the common requirement for cloth is to be soft on touch. This results in featureless object, that is likely hard to recognize

D) One of ways how to attach extension hair is to use specialized glue. The picture does not make clear what product is sold without category name as context

E) Image of a smartphone protection screen contains a smartphone. The instances, where multiple products are depicted, are common.

F) The last image depicts bright tablecloth on a table. The classifier might consider the table as the main object on this photo. It might also be confused by the bright tablecloth.

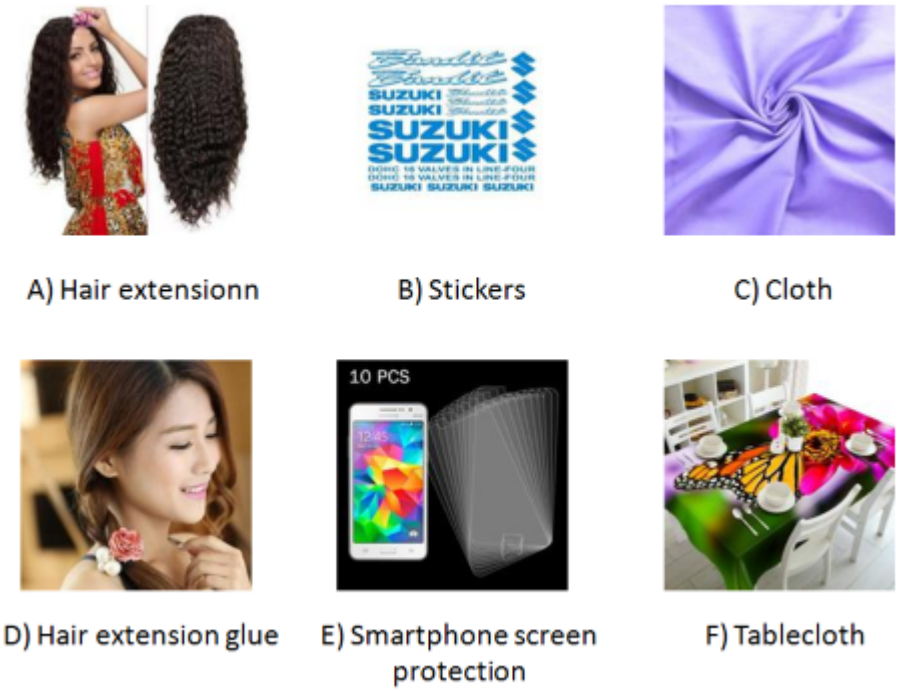


Figure 4.10: Chosen difficult examples.

Chapter 5

Classifier design

5.1 Classification of products

As stated in the previous chapter, each product can have one up to four images. No other metadata than images is present in the dataset. Because dataset is of enormous and end-to-end training would require too much of computational power, transfer-learning with the pre-trained network was chosen. The ImageNet dataset [2] was chosen for transfer-learning, because it's most prominent dataset used for transfer-learning in image classification, with plenty of pre-trained networks available. This design decision restraints classifier to the input of a single image of three channels which brings up two straightforward ways of classification:

- Stack channels of all images into a single image. If the product is missing some channels, represent it with the all-white placeholder. Consider single product transformed into one deep image as a single training example.
- Use every image as a separate training example. Group all predictions belonging to the single product and average them.

The second option was used, as the first option was discarded for following reasons:

- As training end-to-end network would be too expensive on limited hardware that is available, images of channel-size 3 must be used. This means that only 3 of up to 4 images could be used with using only gray-scale representations.
- Most of the dataset consists of products with a single image, which would leave most of the training examples with two unused channels.
- Converting images to gray-scale leads to loss of color information. The importance of color for detection of book covers was demonstrated in [5]. It is believed that color importance of book covers can be generalized on most of the dataset.
- Keeping the input as close to the input format used in the dataset which the network was pre-trained on is a better design choice to utilize as much of learned features as possible.

5.2 Classification metrics

5.2.1 Accuracy scores

Top-1 score are used to measure training and validation split classification performance on training instances, therefore product images. The score was used to be the same as one used in the competition [4]. Score is defined below in formula 5.1, where T_1 is Top-1 score, c_1 is number of training instances where most probable class is the same as target label and n is number of training instances in split.

$$T_1 = \frac{c_1}{n} \tag{5.1}$$

5.2.2 Kaggle score

Kaggle score is a metric assigned by an online system that is part of Kaggle competition framework. In order to obtain the score, one must upload a CSV file containing product identifiers from unlabeled test dataset portion. The score cannot be computed offline since correct labels of test dataset are not publicly available and not available to the author of this work. According to definition on Kaggle [4], the score is number of correctly classified products over all of the products, meaning that it's Top-1 score for products, which is written in form of equation, where K_1 is Kaggle score, p is number of correctly classified products and m is number of products in test dataset.

$$K_1 = \frac{p}{m}$$

5.2.3 Baseline

The baseline is defined by Kaggle competition [4] and is set at 0.51759. The user `inversion` describes used algorithm on behalf of Cdiscount company in discussion post [55]. The user states that he used perceptual hashing algorithm and assigned a class of closes training example. The user links to wikipedia and does not link any concrete implementation or study. The users also states that classification took " 10 hours on a 64-core CPU". A perceptual hashing function is a function generating fingerprint of multimedia (images in this example).

5.3 Optimizer

Adadelta optimizer [30] as implemented in Keras [31] with default values lr=1.0, rho=0.95, epsilon=None, decay=0.0 left intact was chosen. The optimizer was chosen because it's state-of-the-art optimizer with low number of parameter that generally works well on it's default settings. Computational difficulty of the dataset forbids meaningful hyperparameter tuning on single-GPU setup.

5.4 Loss function

All the models are performing classification task into multiple categories and so crosscategorical entropy [25] is used.

5.5 Fine-grain classifier architecture

A fine-grain classifier is a classifier that labels unlabeled product image instances straight into fine-grain level 3 category. The taxonomy is not used and the classifier is relied on to classify into many target variables. The architecture is depicted in figure 5.1. The input of classifier are unlabeled data and output are Level 3 labels.

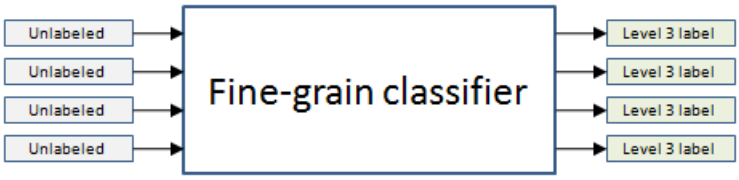


Figure 5.1: Fine-grain classifier architecture schema.

5.6 Hierarchical classifier architecture

This work proposes a hierarchical architecture of deep convolutional neural networks to tackle the complexity of employed dataset. The proposed architecture is composed of a coarse classifier that decides which specific classifier should be used. The main idea of the classification process is following:

1. An unlabeled image of product is passed to coarse classifier
2. A coarse classifier classifies image to one of level 1 categories
3. Based on level 1 category, specific classifier is chosen
4. The chosen classifier classifies image in one of the target classes of level 3 categories

The architecture exploits existing hierarchical taxonomy of product categories that are present in the dataset. Each product in training set is labeled with level three category, for which is known its parent in level two and its parent in the top level. Using top level for first coarse classifier allows employing specific classifiers for certain level one categories or group of categories. Overall architecture can be seen in figure 5.2.

The main advantages of this approach are:

- **Shorter training time**. As shown in 7.6, it's faster to train a classifier that classifies all training examples to 49 categories than to 5270 categories. This allows more rapid prototyping and provides earlier feedback.

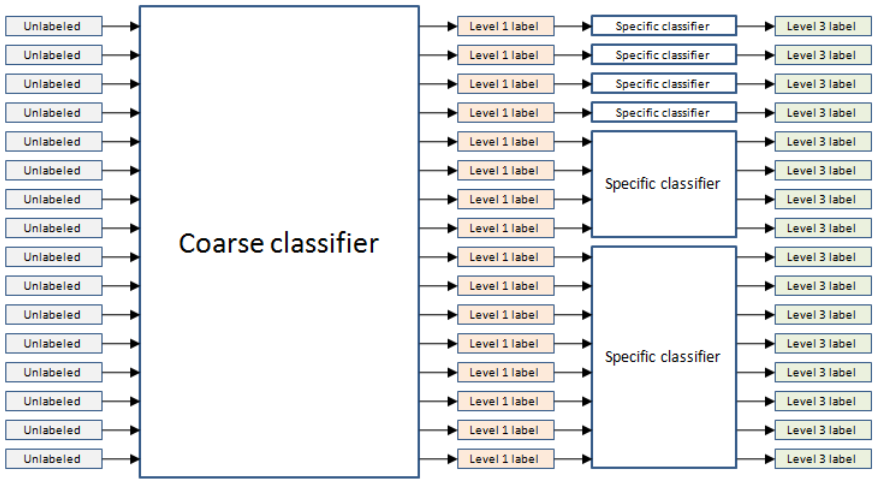


Figure 5.2: Overall architecture schema.

- **Minimizing problems that might occur during training** by using simpler models. Instead of training one very deep model, one can use shallower networks and avoid possible complications, such as vanishing gradients.
- **Modularity**. Specific classifiers can be any classifiers. This allows for different models or network architectures on resulting smaller but more difficult datasets.
- **Transferability of learned specific classifiers**. As shown in 7.10, it is possible to use the specific trained network as a pre-trained network for transfer learning.

The main disadvantages of this approach are:

- **Multiple separate optimizations**. The intuition is that multiple disjoint optimizations tend to perform worse than single joint optimization. This architecture also goes against the main idea of convolutional neural networks, that as many decisions as possible should be learned and made by the network. Although hierarchical architecture did achieve better results as seen in 7.8, it is likely that single end-to-end learned network would overperform it. However, end to end learning was not conducted in this work to verify this.
- **Harder implementation**. The implementation of hierarchical networks architecture is more time consuming than running single training of existing model.
- **More design decisions**. The designer must decide on a strategy how to find out which high-level categories are going to be covered by what models. The designer can then try different models for each category or category group, leading to an explosion of the overall number of decisions.
- **Single point of failure**. The coarse classifier must have good performance, as it affects the operation of all specific classifiers, having a significant impact on overall performance.

5.7 Conversion of fine-grain classifier to coarse classifier

As taxonomy is known and each level 3 label as a link to level 2 and level 1, level 3 label can be easily converted to level 1. This process can convert fine-grain classifier to coarse classifier, allowing to employ architecture described in section 5.6. The architecture can be seen in figure 5.3.

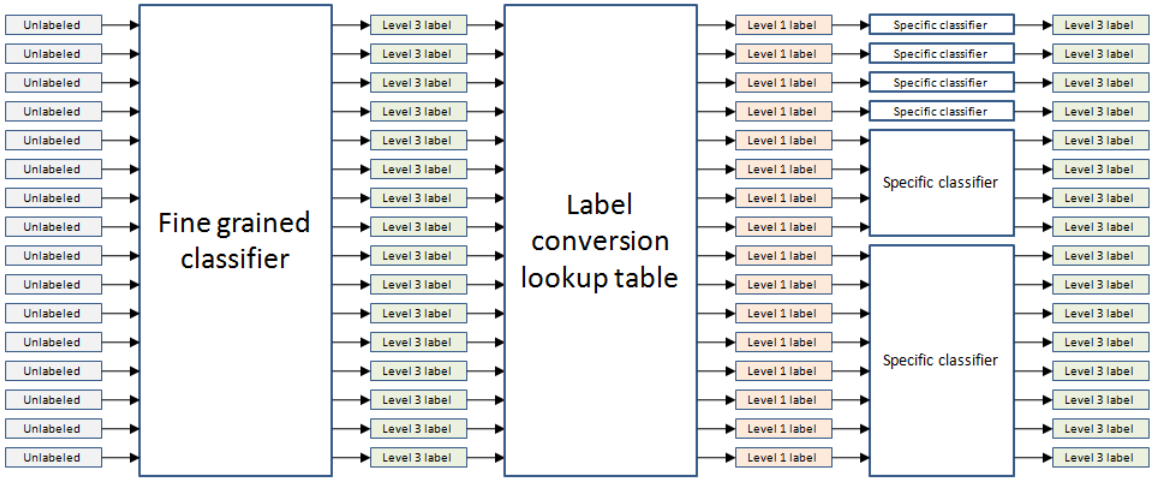


Figure 5.3: Architecture employing conversion from fine to coarse classifier.

5.8 Specific classifiers

The goal is to cover all top-level categories with specific classifiers. The categories with very high accuracy can be trained by a single model, while under-performing categories should be trained in groups to get a chance of correction of possible coarse classifier misclassification. Categories are sorted by their accuracy, with top categories that having accuracy higher than 80% get their own classifier, categories between 60% and 80% form one group and categories with accuracy over 10% form the other group. Categories with less than 10% accuracy were categories with a very small number of examples.

5.9 Additional alternative specific classifiers

Some categories or subcategories might have specific difficulties or features that could be utilized. The exceptionally hard category is `LIBRAIRIE` containing images of book covers.

5.9.1 Mask R-CNN as a feature extractor

Sometimes the product image contains many small objects that together identify the overall product. A good example is a kitchen unit that often contains refrigerator, sink, spoons, oven or coffee maker. Together, these objects likely provide enough of information to identify a

kitchen unit. Similar approach while using YOLO detector [56] was studied in [57]. Mask R-CNN [58] is a powerful object detector with simultaneous segmentation. It's pre-trained Keras implementation [59] detects 81 different object classes. An example of detection can be seen in figure 5.4.

Figure 5.4: Kitchen unit with highlighted Mask-RCNN detections.

5.9.2 OCR for text extraction from book covers

One of the most prominent and discriminating features of a book cover is its name and author. Although text printed on a book cover is very short, it can help to classify book genre in many ways:

- The author is well-known and sticks to a limited set of genres (J.R.R. Tolkien, G.R. R. Martin, ...)
- The book has its genre in the name (ex: "French history", "Taking care of cats", ...)
- The book name is already in dataset

The classification schema is depicted in figure 5.5.

Figure 5.5: Classifying book covers based on OCR text extraction.

Chapter 6

Implementation

6.1 Implementation framework

Keras [31] was used as deep learning framework, with Tensorflow [32] running as a backend. All code was written in Python 3 [54] programming language, with the usage of miniconda [60] environments and popular scientific libraries such as scipy, numpy, and pandas that are part of scipy ecosystem [61].

6.2 Fine-grain Inception V4 transfer learning without fine-tuning

InceptionV4 [48] was chosen from available Tensorflow models in tensorflow models repository [32] on GitHub [1], because at the time of choosing, the model reported the best performance on ImageNet dataset of 80.2 Top-1 accuracy. Note that repository is being continuously updated and that deep learning research field is rapidly evolving.

Transfer learning without fine-tuning does not require a repeated run of convolutional layers. Every time an image is passed through the pre-trained frozen network, the result is the same. Running the whole network while training different classification head would, therefore, pose a significant overhead. This overhead can be mitigated by pre-computing and caching features used as an input of trained classifier.

6.2.1 Dataset preparation

Although InceptionV4 model is not present in Keras framework, it is possible to extract features using Tensorflow and perform classifier learning in Keras, which is what has been done in this work. Inception4 network pre-trained on ImageNet [2] was downloaded from Google's repository and using a custom written script in a Jupyter notebook, 1536 dimensional feature vectors from layer "InceptionV4/Logits/AvgPool_1a/AvgPool" were extracted and stored into small H5 files, called chunks.

The extraction process follows:

[1] https://github.com/tensorflow/models/tree/master/research/slim

1. Products are loaded from dataset one by one in a linear manner using BSON iterator, iterating through BSON file. While loading, images are transformed from JPEG byte array to numpy array of shape $180 \times 180 \times 3$.
2. Each image is then resized to shape accepted by InceptionV4 which is $299 \times 299 \times 3$ using antialiasing.
3. The images are then buffered into batches of 250 items
4. Images are turned into features using `session.run` of Tensorflow on InceptionV4 graph
5. The prediction results of shape 250×1536 are then buffered into array of size 15000×1536 and stored into H5 file resulting in size of 88.0MB
6. Whole dataset is therefore transformed into 471 files having overall size of 40,4 GB

Having dataset split across small H5 files presents a significant advantage over a monolithic H5 file. The dataset can be split into multiple partitions. If a dataset is large (as it is in case of this work), it might not fit into a partition of SSD drive. In this case, chunks that were part of validation split were kept on the partition of slower SSHD drive. The validation split was 80% training split and 20% validation split.

6.2.2 Architecture

The classifier of choice was fully connected network implemented in Keras framework. The network constituted of two fully connected layers, first having 8000 ReLU activations and second having 4000 ReLU activations. The following layer was dropout layer of rather an aggressive dropout 50% leading into classification layer of 5270 softmax activations corresponding to level 3 categories. Adadelta optimizer with and categorical cross-entropy loss was used.

The order of chunks fed to network during each epoch was randomized as well as arrays in chunks shuffled, fulfilling the requirement on shuffling data before each epoch.

6.3 InceptionV4 coarse classifier transfer learning

The same architecture and dataset were used to train a coarse classifier as defined in 5.6 for level 1 categories. The only applied changes were: the change of the last layer in the classifier to 49 softmax activations instead of original 5270 to match the number of level 1 categories. The dropout was significantly reduced to 12.5% since the model was considerably smaller while dealing with the same number of instances.

6.4 Hierarchical architecture

Classification pipeline is split into two steps in order to classify products using a hierarchy of models. Firstly, coarse classifications are computed. Secondly, specific classifiers apply their predictions.

1. A coarse classifier classifies all products into level 1 categories
2. Level 1 categories are then converted into any of corresponding level 3 categories
3. Submission file containing product id and assigned label with is written

This process creates very bad predictions, as the coarse classifier determines level 1 category and randomly picks level 3 label. This means, that if a product is a book, it will be hopefully classified as a book, but then genre of the book will be picked at random. The specific classifiers are then employed to correct these initial guesses. Following steps are performed for each specific classifier:

1. A specific classifier is loaded
2. Submission file created in the previous step is loaded
3. All products that do not belong to the specific category or set of categories learned by the specific classifier are filtered out
4. All remaining products are classified
5. New submission file is stored, ready to be loaded and processed by another specific classifier

This approach is specific to the Kaggle competition setup and allows to submit files separately or improve previous classifications. Different implementation approach would be chosen, should hierarchical architecture be used in the production environment of an actual E-commerce site.

6.5 ResNet50 fine-grain transfer learning with fine-tuning

Pre-trained ResNet50 network from Keras [31] was used. The network was pre-trained on ImageNet dataset [2]. The ResNet architecture was chosen because there is very nice Keras implementation [59] that allows to build custom architecture. Although conducted experiments were not used in this work, the design decision to use ResNet in order to compare end-to-end trained ResNet18 with other networks, the design decision was kept as some specific models were already trained and hierarchical architecture was meant to be used with single architecture so that results are comparable.

6.5.1 Dataset preparation

The training instances are loaded directly from dataset file using random access reading. Each instance was read and then fed to parallel processing pool of four threads that decoded JPEG bytes and preprocessed image as described in the following section. The label is converted into one-hot representation. Images were then collected into batches of size 500 instances, that was found to yield best training speed on used ResNet50 models with using `GTX 1080 TI` GPU having 11 GB of memory. The whole pipeline was implemented as a chain of modular python generators:

- yield tuple of JPEG bytes, label, and product id
- feed images to parallel pool and then yield image, label, and product id
- yield batches of lists of items
- convert batches of lists into numpy arrays and yield them

6.5.2 Image preprocessing

Because dataset is very large, no image augmentation is used. Moreover, most of the images are presented in a single orientation (Book, CD, DVD, ..) as most products have well defined upright orientation in which they are photographed. The image preprocessing pipeline follows:

- The mean image is subtracted from an input image
- Image intensities are rescaled
- An input image is padded with white pixels from all sides to enlarge it to the size of $198px \times 198px$ in order to use it with a pre-trained ResNet50 network.

6.5.3 Architecture

The bottom dense layer of ResNet50 pre-trained model was replaced with a dense layer of 5270 softmax activations. This means that replaced layer was connected to the output of the GlobalMaxPooling2D layer. All layers above layer 159 were frozen, therefore their weights were not subjected to learning, all layers below layer 159 were subjected to training, resulting in 16,318,614 training parameters. The model was compiled with categorical cross-entropy loss and Adadelta [30] optimizer with default parameters recommended by Keras documentation [31]. The overview of architecture can be found in table 6.2.

	Number of trainable	Number of non-trainable	Total
Layers	16	158	174
Parameters	16,318,614	18,067,328	34,385,942
Loss	Crosscategorical entropy		
Optimizer	Adadelta		

Table 6.1: Summary of ResNet50 fine-grain classifier

6.6 ResNet50 coarse transfer learning with fine-tuning

The same network as in the previous section was used, with same dataset preparation and image preprocessing. The only difference was the number of activations in the bottom layer, which has been changed from 5270 activations to 49 activations. The labels of dataset were transformed to top-level category labels. The architecture is characterized by table 6.2.

	Number of trainable	Number of non-trainable	Total
Layers	16	158	174
Parameters	16,318,614	18,067,328	34,385,942
Loss	Crosscategorical entropy		
Optimizer	Adadelta		

Table 6.2: Summary of ResNet50 coarse classifier

6.6.1 Specific classifiers

The specific classifiers are using the same network as in previous sections, with a different number of activations and trainable parameters. Summaries of all networks are captured in table 6.3. The stopping criterion is that training is to be stopped as soon as the network stops yielding significant validation score improvement. With many models to train, it's better to start training new model of a different category (or category group) as that improves overall hierarchical model score faster. However, no early stopping callback was used, as epoch times were around one hour and easily terminated by the user.

Description	Hyperp.	Number of trainable	Number of non-trainable	Total
Books	Layers	41	133	174
	Parameters	16,163,746	7,755,904	23,919,650
	Categories	162		
	Loss	Crosscategorical entropy		
	Optimizer	Adadelta		
AUTO MOTO	Layers	41	133	174
	Parameters	16,733,368	7,755,904	24,489,272
	Categories	440		
	Loss	Crosscategorical entropy		
	Optimizer	Adadelta		
MUSIQUE	Layers	41	133	174
	Parameters	15,891,229	7,755,904	23,647,133
	Categories	29		
	Loss	Crosscategorical entropy		
	Optimizer	Adadelta		
TELEPHONIE	Layers	41	133	174
	Parameters	16,005,973	7,755,904	23,761,877
	Categories	85		
	Loss	Crosscategorical entropy		
	Optimizer	Adadelta		
ACCESSOIRES	Layers	41	133	174
	Parameters	15,899,425	7,755,904	23,655,329
	Categories	33		
	Loss	Crosscategorical entropy		
	Optimizer	Adadelta		
INFORMATIQUE	Layers	41	133	174
	Parameters	16,112,521	7,755,904	23,868,425
	Categories	137		
	Loss	Crosscategorical entropy		
	Optimizer	Adadelta		
Category group 1	Layers	41	133	174
	Parameters	20,002,197	8,611,712	28,613,909
	Categories	2453		
	Loss	Crosscategorical entropy		
	Optimizer	Adadelta		
Category group 2	Layers	33	141	174
	Parameters	20,002,197	8,611,712	28,613,909
	Categories	1841		
	Loss	Crosscategorical entropy		
	Optimizer	Adadelta		

Table 6.3: Summaries of specific category classifiers

	category	lvl1_acc	mean_lvl3_acc
0	CHAUSSURES - ACCESSOIRES	0.935016	0.231398
1	LIBRAIRIE	0.930811	0.112841
2	MUSIQUE	0.912822	0.105414
3	TELEPHONIE - GPS	0.909753	0.304265
4	BIJOUX - LUNETTES - MONTRES	0.893552	0.327899
5	INFORMATIQUE	0.873011	0.277602
6	AUTO - MOTO	0.862579	0.282801
7	LITERIE	0.843319	0.274700

Table 6.4: Table listing categories with individual models

6.6.2 Individual categories

The table below 6.4 shows categories for which independent models were fitted. The difference between fine-grain ResNet50 6.5 model's ability to recognize top level category is high, while not being able to correctly classify specific level 3 categories.

6.6.3 Category group 1

The table bellow 6.5 shows categories included in group 1 and validation Top-1 score of fine-grain ResNet50 6.5 classifier which was used to determine the split. The difference between level 1 accuracy and mean accuracy of level 3 categories shows potential of improvement.

6.6.4 Category group 2

The table bellow 6.6 shows categories included in group 2 and validation Top-1 score of fine-grain ResNet50 6.5 classifier which was used to determine the split. The difference between level 1 accuracy and mean accuracy of level 3 categories shows potential of improvement.

6.7 Alternative methods

6.7.1 Using Mask-RCNN as a feature extractor

Keras implementation of Mask-RCNN with pre-trained model can be found on [62] and have been used to perform feature extraction. The first step of the implementation is to extract features. Following features were chosen for each detection:

- top category id of detection
- probability of top category
- mask area, computed as number of mask pixels over area of bounding box
- mask bounding box, computed from mask

The obtained results were collected into batches of lists and stored using pickle python serialization library in separate files.

	category	lvl1_acc	mean_lvl3_acc
23	JEUX VIDEO	0.593011	0.170330
24	SPORT	0.549653	0.116287
25	ART DE LA TABLE - ARTICLES CULINAIRES	0.543822	0.085995
26	MERCERIE	0.529655	0.183065
27	PUERICULTURE	0.478841	0.100928
28	SONO - DJ	0.457444	0.141257
29	JARDIN - PISCINE	0.451209	0.131998
30	ANIMALERIE	0.432689	0.103930
31	INSTRUMENTS DE MUSIQUE	0.432429	0.139480
32	LOISIRS CREATIFS - BEAUX ARTS - PAPETERIE	0.406510	0.124570
33	EPICERIE	0.405018	0.085615
34	BATEAU MOTEUR - VOILIER	0.380626	0.134984
35	MANUTENTION	0.345398	0.155389
36	PARAPHARMACIE	0.334813	0.054902
37	DROGUERIE	0.325758	0.068038
38	TENUE PROFESSIONNELLE	0.317406	0.141960
39	CONDITIONNEMENT	0.296374	0.148985
40	ELECTRONIQUE	0.253662	0.103178
41	POINT DE VENTE - COMMERCE - ADMINISTRATION	0.238457	0.068403
42	MATERIEL DE BUREAU	0.192859	0.102456
43	FUNERAIRE	0.117647	0.117647

Table 6.5: Description of category group 1

	category	lvl1_acc	mean_lvl3_acc
8	TATOUAGE - PIERCING	0.805052	0.244553
9	BAGAGERIE	0.797549	0.204752
10	DVD - BLU-RAY	0.792854	0.158592
11	MEUBLE	0.769749	0.216701
12	PHOTO - OPTIQUE	0.763408	0.181480
13	DECO - LINGE - LUMINAIRE	0.760144	0.223677
14	COFFRET CADEAU BOX	0.745098	0.165409
15	ELECTROMENAGER	0.741398	0.149360
16	HYGIENE - BEAUTE - PARFUM	0.711315	0.122438
17	VIN - ALCOOL - LIQUIDES	0.693420	0.170946
18	JEUX - JOUETS	0.686930	0.117135
19	TV - VIDEO - SON	0.667777	0.162798
20	BRICOLAGE - OUTILLAGE - QUINCAILLERIE	0.631755	0.191393
21	MATERIEL MEDICAL	0.629986	0.089270
22	ARTICLES POUR FUMEUR	0.624543	0.117827

Table 6.6: Description of category group 2

6.7.2 Book OCR classifier

The book cover images were extracted from the dataset into a directory as JPEG images. Images were then loaded in Matlab 2016a and text was extracted using OCR implementation present in Matlab [63]. OCR results were stored in JSON file and analyzed using python. All non alpha-numeric characters were discarded using regular expression [Â-Za-z0-9
s]+[2] and the text was converted to lower case. The text was then tokenized by spaces, resulting in a set of tokens for each book cover. All book covers were split into 80% training and 20% testing split. The extracted tokens were aggregated to their respective categories. Since extracted texts from covers were very short, no stemming or lemmatization was used. The stopwords were not dropped. The classification was performed in form of binary matching with the category having the most matched tokens being a winner.

[2]Regular expression that matches all non-alphanumeric characters with exception of whitespace characters was used to replace matched substring with the empty string, thus removing matched characters

Chapter 7

Experimental evaluation

7.1 Experimental setup

Experiments were conducted on a personal computer equipped with Intel Core™i3-4170 CPU @ 3.7GHz with two physical and four logical cores, 16 GB of DDR3 RAM, 256GB SSD hard drive, 500GB SSHD hard drive and single GTX 1080 TI GPU with 11GB of memory.

7.2 Experiments overview

Experiment	Implementation	Design	Top-1 score
7.3	InceptionV4 6.2	fine grain classifier 5.5	0.4080
7.4	InceptionV4 6.3	corase classifier 5.6	0.6852
7.5	ResNet50 6.5	fine grain classifier 5.5	0.5627
7.6	ResNet50 6.6	coarse classifier 5.6	0.7940
Specific classifiers 7.7	ResNet50 6.6.1	specific classifier 5.8	various
Overall model 7.8	Hierarchy of ResNet50 6.6.1	hierarchical classifier 5.6	0.61061[a]
Book classifier 7.10	ResNet50	Pre-trained on AUTO-MOTO	0.2318
Book classifier 7.10	ResNet50	Pre-trained on ImageNet	0.2571
Book classifier 7.10	ResNet50	Pre-trained on books	0.2824

Table 7.1: Overview of conducted experiments

[a]Score evaluated by Kaggle over products, not product images

7.3 InceptionV4 fine-grain classifier

This section concerns results of fine-grain classifier as designed in 5.5 with implementation described in 6.2.

7.3.1 Results

The training took little over 18 hours before starting to plateau. The training was manually interrupted in favor of running different experiment after presenting itself with unsatisfying results. The model reached training Top-1 accuracy of 44,85% and validation Top-1 accuracy of 40,80% as can be seen in figure 7.1. The training was interrupted and resumed multiple times to make hardware available for other experiments, which can be seen in the figure in the form of discontinuities and different colors. The significant discontinuity between green and gray run on Sunday 26 is believed to be caused by moving a large portion of the training set to SSD drive, which in turn sped up training.

7.3.2 Discussion

The poor performance of transfer-learning without fine-tuning can be explained by the large disparity of ImageNet data and product images. The InceptionV4 network was fitted to recognize a large number of classes that are not useful in product classification, such as cars, animals or food. Hyperparameter tuning is unlikely to increase accuracy by a significant margin.

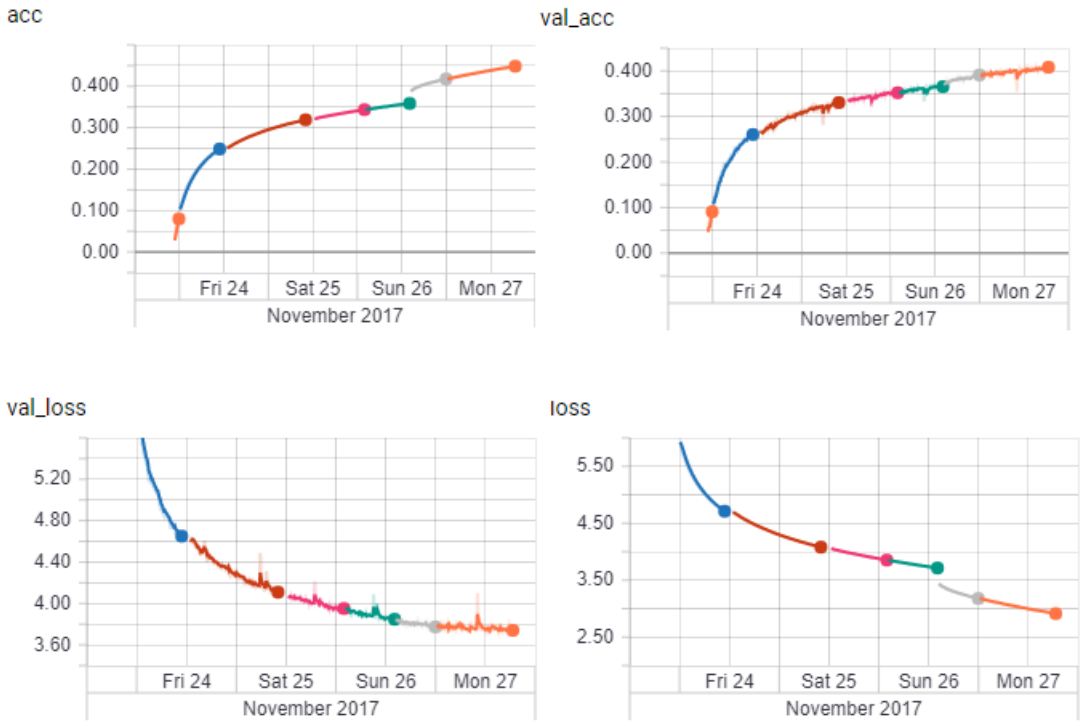


Figure 7.1: Progress of InceptionV4 transfer learning

7.4 InceptionV4 coarse classifier

The training took one day and 17 hours reaching the accuracy of 58,57% on validation split and 68,52% on training split. The progress of training can be seen in figure 7.2.

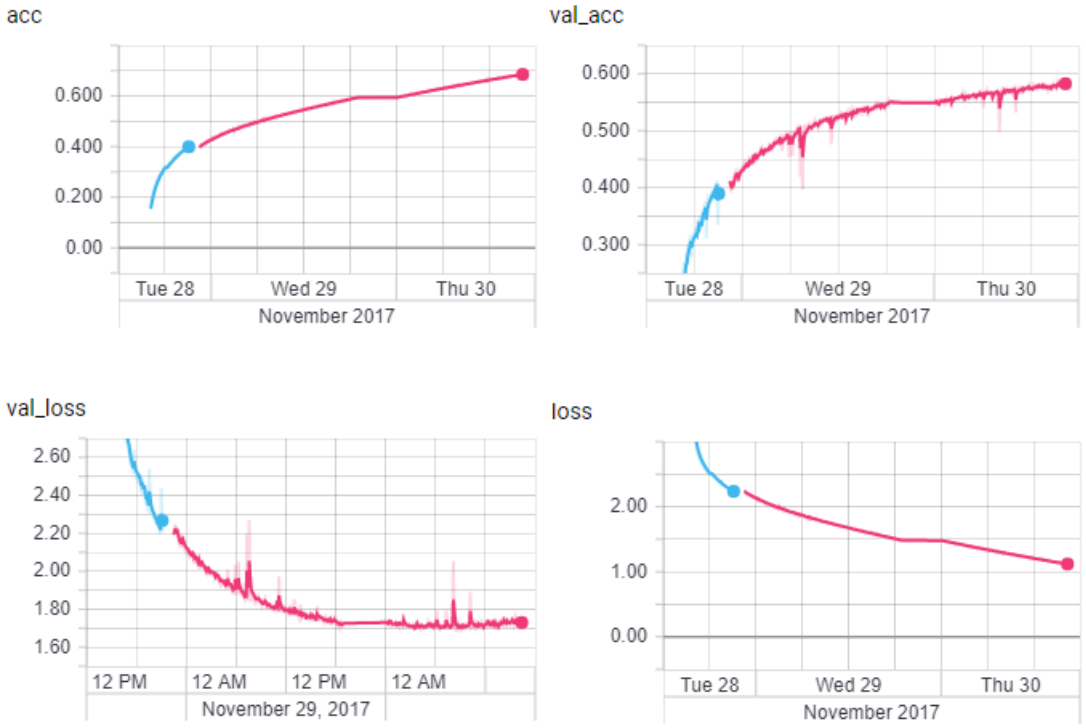


Figure 7.2: Progress of InceptionV4 transfer learning of coarse model

7.5 ResNet50 transfer-learning fine-tuned fine-grain classifier

The result can be seen in table 7.2. The training was manually interrupted after first epoch that reported validation score decrease. Training continuation with lower learning rate from previous check-pointed epoch would likely not bring an improvement worth of computational time. The training curves can be seen in figure 7.3.

Batch size	Epoch	Training time	Training Top_1 score	Validation Top_1 score
500	3	22h 12m	0.5692	0.5627

Table 7.2: ResNet50 transfer-learning fine-tuned fine-grain classifier results

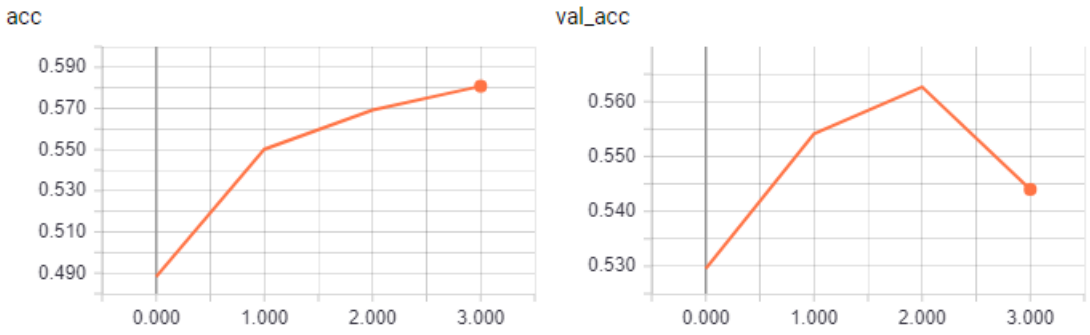


Figure 7.3: Progress of ResNet transfer learning with fine-tuning on bottom categories

7.6 ResNet50 transfer-learning fine-tuned coarse classifier

The result can be seen in table 7.2. The training was manually interrupted after first epoch that reported validation score decrease. Training continuation with lower learning rate from previous check-pointed epoch would likely not bring an improvement worth of computational time. The training curves can be seen in figure 7.4.

Batch size	Epoch	Training time	Training Top_1 score	Validation Top_1 score
500	3	1d 11h 35min	0.8214	0.7940

Table 7.3: ResNet50 transfer-learning fine-tuned coarse classifier results

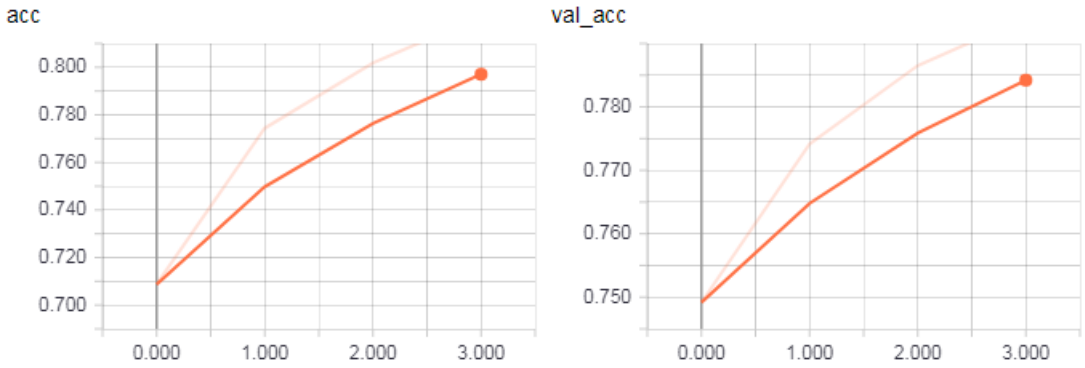


Figure 7.4: Progress of ResNet transfer learning with fine-tuning on top level categories

7.7 Specific classifiers

Tabular schema of specific classifier results follows in table 7.4. All classifiers had the batch size set at 500. Training was stopped manually after first epoch that reported validation

score decrease. Instead of fine-tuning the model, training of another model has been started in favor of better overall hierarchical model score improvement. The number of parameters of each model was kept around 16 million by adjusting the number of frozen layers.

Classifier	Epoch	Training Top_1 score	Validation Top_1 score
Books	3	0.4989	0.3847
AUTO-MOTO	8	0.9188	0.8229
MUSIQUE	2	0.4344	0.4019
TELEPHONIE	7	0.9484	0.8943
ACCESSOIRES	13	0.9575	0.7467
INFORMATIQUE	7	0.9259	0.8459
Category group 1	3	0.6451	0.5027
Category group 2	3	0.5911	0.5777

Table 7.4: Specific classifier results

7.8 Performance of overall hierarchical architecture

The overall reached Kaggle score based on late submission was 0.61061. The Kaggle score measures Top-1 accuracy on products.

7.9 Alternative classifiers

7.9.1 Using Mask-RCNN as a feature extractor

The experiment was prematurely interrupted and not finished, as feature extraction took too much time. Extraction for book dataset containing almost 800,000 images would take several days. This yielded using Mask-RCNN as a feature extractor as an impractical idea.

7.9.2 Using OCR to judge book covers

Only 30% of book covers yielded non-empty OCR recognition, which is due to the nature of some book covers as can be seen in figure 7.5 and low image resolution. Meaningful OCR recognitions make a very small fraction of all recognitions, with most non-empty recognitions being scrambled letters such as "1 qu mh q c i 5". Good recognition came from book covers with big clear fonts and text characterizing category, such as "Shakespeare" or "Histoire" as seen in figure 7.6. The overall result was test accuracy of 4,22%. No further experiments were conducted, as dataset resulting from OCR is of rather lower quality.

7.10 Transfering learned knowledge of specific classifier

One of specific classifiers specializes to classifying books into 162 categories of genres. As mentioned in chapter 2, there is a study [5] that deals with classifying books into 30 genre categories with a publicly available dataset. The hypothesis stated in 5.6, that specific classifier could be transferable to other dataset is verified in this section.

Figure 7.5: Book covers that yielded empty OCR text

Figure 7.6: Book covers that were correctly classified based on OCR

7.10.1 Experiment details

The dataset was obtained from [6] and contains 51,000 book covers in training split and 5700 files in test split. The size of images was transformed to the smaller size of 198×198px to fit input size of the transferred model. The last layer of trained book classifier model was removed and all other layers were frozen. Dense layer of 30 softmax activations was then added to the model, resulting in a classifier. The last layer was trained with cross-categorical entropy loss and Adadelta optimizer. The reduce learning rate on plateau callback with the patience of 2 epochs, factor set 0.2 and minimum learning rate to 0.0001 was used. The early stopping criterion callback with minimum delta = 0 and patience of 4 epochs was added. Training was performed on batches of 500 images. All experiments were conducted in the same manner, with the same parameters on the same hardware.

Three experiments were conducted:

A Transfer learning of ResNet50 model with ImageNet pre-trained weights as a baseline

B Transfer learning of ResNet50 model with weights pre-trained on book category

C Transfer learning of ResNet50 model with CDiscount AUTO-MOTO category pre-trained weights (classifier trained in this work) that is not relevant to books

With following hypothesis:

H1 Model B will over-perform models A and C, because it has learned book-specific features

H2 Model C will underperform model A, because it has learned more specific features than ImageNet pre-trained model, that was trained on more general dataset

7.10.2 Results

Results can be seen in table 7.5 bellow. Learning progress can be seen in figure 7.7. The dark blue color is ResNet50 pre-trained on books. The red is ResNet50 pre-trained on ImageNet and the lightblue is ResNet50 pre-trained on irrelevant AUTO-MOTO category.

Classifier	Epoch	Training Top-1 score	Validation Top-1 score
ResNet50 - ImageNet	14	0.3235	0.2571
ResNet50 - books	20	0.3392	0.2824
ResNet50 - AUTO-MOTO	15	0.2817	0.2318

Table 7.5: Result of experiments of transfer-learning fitted features.

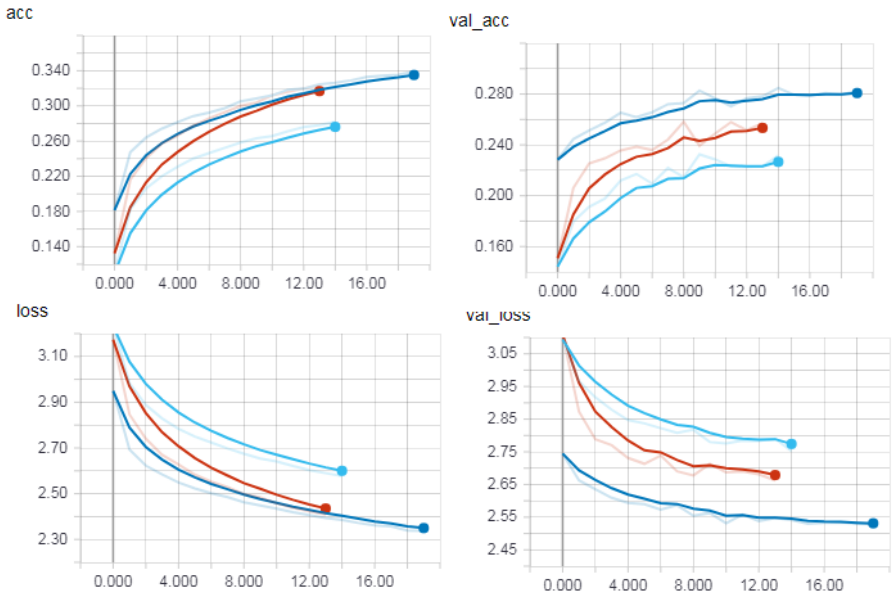


Figure 7.7: Plot showing learning progress of all three networks combined.

Chapter 8

Discussion of results

8.1 Discussion of evaluated models

8.1.1 Transfer learning without fine-tuning convolutional layers

Both fine-grain and coarse models as defined in chapter 5 were implemented as InceptionV4 [48] transfer-learned models from bottleneck layer features stored on disk.
Advantages:

- The approach reduced computational time by eliminating forward pass through convolutional layers of InceptionV4 model by precomputing results and storing them on disk
- The dataset was stored in many small H5 files as described in 6.2.1, which allowed to split the dataset across multiple hard-disk partitions

Disadvantages:

- Fully connected layer has added computational complexity, mitigating advantage achieved by cutting-off forward pass
- The models did not converge above the set benchmark

Future work is described in section 8.4.11 as it cites approach of different Kaggle competitor.

8.1.2 Transfer learning with fine-tuning some convolutional layers

The ResNet50 [3] was used by directly feeding input images to an optimization algorithm, allowing to re-train some chosen layers that were initially trained on ImageNet dataset [2]. This approach led to significant performance boost over the one described in the previous section. The model has surpassed baseline of 0.51759 Top-1 score on products and achieved 0.5627 Top-1 score on product images.
Advantages:

- The approach allows to include and exempt layers from training at will[1]
- The model achieved better than baseline performance

Disadvantages:

- The model learns for long time

Future work could bring progress in experimenting with enabling/disabling layers after each epoch or between batches similarly as done in network Stochastic depth network [43]. Different optimization function or loss function could contribute to better convergence.

8.1.3 Hierarchical model

The hierarchical model consisting of multiple ResNet50 networks with some fine-tuned convolutional layers have shown the best results, outperforming baseline by almost 10% on Top-1 score. Instead of semi-randomly tweaking hyperparameters to obtain a better result, the bad performance of network was analyzed. By comparing performance between top level and bottom categories, it was found out that many top-level categories have a good performance of Top-1 score $> 80\%$, while the mean of level 3 category accuracies within those top-level categories was very low. This means, that while the network has learned very well to recognize the book, it could not shelve it under the correct genre. Therefore, the expected improvement of the model is soundly motivated and consequently experimentally confirmed. The discussion of experimental evaluation of the hypothesis that networks trained on specific categories learn more specialized features is in section 8.2.1. Advantages and disadvantages of architectural design are stated in section 5.6.

8.1.4 Choosing group of categories for specific classifiers

The level 1 categories are grouped according to the accuracy of a coarse classifier. The better possibility might be to group categories based on the confusion matrix, that can reveal relationships between categories. Two categories that are confused between each other could be resolved by one specific model learned to discriminate training examples from those categories.

8.2 Comparison with state-of-the-art

To the authors best belief, there are not many studies concerning the classification of products based solely on images, that would deal with a dataset of comparable size. The only large-scale recognition study found was [16]. The authors of the study themselves complain about the absence of large-scale studies. The study did not release their dataset, so comparison of obtained results was not possible. However, comparison with Kaggle competitors that solved the exact same task follows in section 8.3. The performance of specific classifiers employed to classify books was compared to the study [5] in the following section.

[1] Not while training

8.2.1 Book cover classifier

One of the specific classifiers trained in hierarchical model is dedicated to "LIBRAIRIE" level one category containing books. Previously mentioned study [5] solves the same problem, albeit on a smaller scale. The dataset used in this work contains over 800,000 book images, while compared study uses only 75,000. Moreover, study categorizes books only into 30 categories, but this work trains classifier distinguishing 162 categories of books. Since the dataset used in the study is publicly available, comparison of the classifier trained n this work was conducted.

In the section 7.10, the specific classifier dedicated to books was retrained on the dataset used in study [5] by training only classification head (single layer of 30 softmax activations) and preserving fine-tuned residual features. The experiment has shown that classifier trained in this work was able to successfully retrain and over-perform retraining from ImageNet dataset and result reached by the study [5] by training only the last layer, while the study used the full capacity of VGG network. Brief result comparison can be found in table 8.1 and more details in section 7.10.

Classifier	Validation Top_1 score
VGG - ImageNet - from [5]	0.247
ResNet50 - ImageNet	0.2571
ResNet50 - books	0.2824

Table 8.1: Result comparison on book dataset [6]

The experiment has also shown that specific classifiers not only gain domain-specific knowledge but also lose their ability to generalize, as a classifier that was fine-tuned to recognize AUTOMOTO category have underperformed ImageNet transfer learning.

It is likely that hyperparameter tuning and using the capacity of the whole ResNet50 network would further increase the score. End-to-end learning from scratch would likely learn very specific features for books, boosting performance even further. Merging the two datasets would likely result in further improvement of the classifier.

8.3 Kaggle competition

The dataset used in this work was part of a Kaggle competition [4], that was running at the time of preparation of this work. Single submission produced by an unfinished model was made at the time of competition deadline, resulting in a score of 0.53691 and the 284th place among 627 submitters. The submission was only 7 places above competition baseline, set at 0.51759 and described as "Image Similarity Benchmark". Since Kaggle allows late submission that does not change leaderboard, the resulting model could be evaluated, reaching 0.61061 Top-1 score on products. Overview can be found in table 8.2.

Name	Top_1 score on products assigned by Kaggle
Image similarity benchmark	0.51759
Leaderboard score	0.53691
Late submission score	0.61061
Competition winner	0.79567

Table 8.2: Kaggle score overview

Username	Place	Accuracy	GPU,RAM	Model
bestfitting	1	0.79567	4x1080	described in 8.4.5
Heng CherKeng	2	0.79352	4x1080	ensemble, see 8.4.6
Azat Davletshin	8	0.77883	2x1080, 64GB	SE-ResNet-101
n01z3	9	0.77735	4x1080	ResNext-101
Vladimir Iglovikov	12	0.77409	4x1080, 32GB	Resnet 50, 101 and 152
Radu Stoicescu	26	0.75328	1x1070	InceptionResNetV2, trained for 25 days
NighTurs	47	0.73938	1x1050, 8GB	ensembe of FC, described in 8.4.9
KoustubhSinhal	51	0.73299	unknown	Inception V3 with focal loss

Table 8.3: Overview of Kaggle contestants solutions

8.4 Comparison with other Kaggle competitors

Description of Kaggle competitors solution and discussion follows. Each section starts with a username of Kaggle user, leaderboard placing, and short summary.

8.4.1 Summary

Most of the users that ranked high had powerful hardware at their disposal, with the exception of Radu Stoicescu, who left training running for 25 days and NighTurs who used very inventive and original approach with respect to others. Mentioned users used less powerful hardware than used in this work while achieving significantly better scores. The overview can be seen in table 8.3.

8.4.2 Using multi-images

The most prominent idea of high-ranked competitors was creating ensembles of end-to-end trained models that classified the product as single training example, classifying all four images at once (if the product had four images). The idea of using a single representation of product was mentioned and dismissed in section 5.1 as it would require training end-to-end network, which is impractical to do on single GPU at this scale of the dataset. However, as described in 8.4.9, the user NighTurs came up with a representation of single product using concatenation of concatenated bottleneck layer features fed to fully connected network classifier, which didn't occur to the author of this work. NighTurs and Heng CherKeng have independently cited the same study concerning recognition of 3D shapes "from a collection of their rendered views on 2D images" [64]. The study describes how to detect a single

object using it's multiple images, which is the case of products in the dataset used in the competition.

8.4.3 Concatenating bottleneck features

Using concatenated bottleneck features is another recurring theme among solutions of high-ranking contestants. The bottleneck features are activations obtained from bottleneck layers that feed into classification head. The users Heng CherKeng, Lam Dang and NighTurs have used some variation of said technique.

8.4.4 OCR to detect text on CD's and books

Using OCR to tackle books and CD's was attempted by multiple users, but the only winner, the user bestfitting was satisfied with his result, albeit straggling with the task. The user lyakaap stated[2] that "Google Cloud Vision API can detect texts from Cdiscount images successfully".

8.4.5 bestfitting, #1, ensemble, multi-images and OCR

The user argues that using models pre-trained with ImageNet is not sufficient since ImageNet has only 1000 classes, meaning that "there will be some representation bottleneck". To resolve this problem, the user adds 1x1 kernel convolutional layer after layer 4 of ResNet34 that he uses for preliminary experiments to increase the number of channels from 512 to 5270 (number of categories in the dataset). With prepared learning rate plan to set specific learning rate at specific epochs, the user obtains a score of 0.72 on end-to-end learning from scratch.

After experimenting with smaller network ResNet34, the user tried to fine-tune models ResNet50, ResNet101, ResNet152, InceptionResNetV2 and InceptionV4. He doesn't state whether he used image-net pre-trained models or trained from scratch. He also reports dissatisfaction with InceptionV4 stating that "Inceptionv4 is a little weak at this stage, others could easily get a score $>$0.755".

The user emphasizes the importance of using all images at once in single optimization. He trained separate models for products with one, two, three and four images, creating an ensemble of them. The model works on channel-wise concatenated images, meaning, that model accepting 4-image products accepts input images with 12 channels. The user calls this approach "multi-image".

The user, similarly as user Lam Dang and this work reports usage of OCR for tackling the complexity of the book category. He reports that he was unable to extract meaningful text as he has not found OCR suitable for French language. Nevertheless, as a last resort, he "extracted the features of boxes from CRNN" and fed them into his "multi-input CNN" which have yielded for him significant performance boost in late stages of the competition.

The user also mentions, that in order to feed images fast enough into his 4x1080Ti setup, he bought multiple 512GB SSD disks.

[2] https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/45728#257868

8.4.6 Heng CherKeng, #2

The user Heng CherKeng of team Convoluted Predictions reports employing an ensemble of models using concatenated bottleneck features. The bottleneck features were obtained by fine-tuning various end-to-end learned networks such as Xception, SE-ResNet101, and others. The bottleneck features are then used as input into a fully connected network. The reader is encouraged to read details in elaborate discuss forum post[3].

8.4.7 Lam Dang #7, ensemble of models

The team of user Lam Dang states training baseline models "InceptionResnetv2, Resnet101, SE-InceptionV3, Xception" [65] with reported accuracy "from .69+ to .72+". They do not state whether they trained from scratch or used pre-trained models. By extracting bottleneck layer features for each image they have constructed vector space on top of which they built additional classifiers:

- RNN with LSTM
- fully connected network
- NetVlad based on [66] and [67]
- LOUPE feature extractor [68]

The overall performace of resulting ensemble is 0.78178. The user also reports trying OCR with no signification performance boost, as similarly reported in this work in section 7.9.2, albeit the user reports more elaborate OCR pipeline with text box extraction and "bag of char ngram" model.

8.4.8 Radu Stoicescu, #26, single GTX 1070 solution

The user stated using InceptionResNetV2 [69] pre-trained model and run training for 10 epochs, reporting each epoch taking about 2.5 days. The user achieved 0.75328 accuracy. No additional details on how he conducted his experiment was provided.

8.4.9 NighTurs #47, single GTX 1050 Ti solution

User NighTurs have been able to achieve a better result on considerable less powerful hardware, using GTX 1050 Ti GPU and only 8GB of RAM. The user reports extracting features from bottleneck layers of ResNet50 and VGG16 models pre-trained on ImageNet and storing them in files. The user emphasizes that feature extraction took over 6 days. Then the contestant trained an ensemble of 44 fully-connected neural network classifiers resulting in the score of 0.73938. The user was kind to publish his work in repository [70]. The user is classified as "competition expert" in Kaggle.

[3] https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/45709

8.4.10 Comparison of this work with other contestants

High ranking contestants used significantly more powerful hardware, with the exception of experienced Kaggle competitor NighTurs who had a very inventive idea and the user Radu Stoicescu who left the training running for 25 days, which was not done in this work as it would be at the expense of other experiments.

Using all product images at once as an input of model was considered and disregarded for performance reason, while users with powerful hardware were able to use the advantage of it. The high-ranking competitors used ensembles of large models such as ResNet152, while this work has used ResNet50 with most of its layers exempted from training. Instead of using ensembles of models, hierarchical model reducing the problem to a manageable size was used. However, OCR to improve book cover classification was used with very similar success as other competitors, which was with exception of the competition winner, none.

In discussion thread [71] some users mention an interest of creating a hierarchical model in a similar way as done in this work, in order to reduce computational complexity, however, no attempts of implementation are documented.

Note that as stated in [72], the Cdiscount company will release study describing and releasing dataset with discussions of the approach taken by top competitors.

8.4.11 Future work based on insights from other Kaggle competitors

The review of approaches of other Kaggle [4] competitors has brought some important insights into possible improvements of presented methods.

- Instead of training single model transfer-learned from features extracted to files, ensemble of weak models should have been used
- Instead of using images as separate training instances, the training instances should have contained all images or their representation in some reduced space
- Using OCR to help with book cover classification was not a bad idea but should have been applied more carefully and in the later stage of model training

Chapter 9

Conclusion

The aim of the thesis was to propose, implement and experimentally evaluate deep neural network architecture for classification of products from images. Single model solution 5.5 and hierarchical architecture 5.6 were proposed in chapter 5, their implementation, including image pre-processing and dataset preparation pipeline, was described in chapter 6 and their thorough experimental evaluation presented in chapter 7. The implementation was done in python [54] while using Keras machine learning framework [31] with TensorFlow computational framework backend [32].

In order to solve the problem, related literature was researched, analyzed and described in chapter 2. Techniques and models used in this work were described in chapter 3. For getting insight into dataset, the chapter 4 have presented thorough analysis, revealing significant challenges. The performance comparison with the state-of-the-art in product classification was discussed in 8.2.

This work concerned solving real world hard classification task of classifying non-food E-commerce products into 5270 categories using only images. The big dataset of size 58,1GB presents a challenges of judging books by their covers, distinguishing smart-phone covers from smart-phones and learning from 12 million images.

The specific classifier solving sub-problem of classifying books trained in this work was compared to [5], surpassing it's Top-1 score of 0.247 by achieving Top-1 0.2824 score. The experiment shown, that features learned on books in dataset used in this work were transferable to different dataset, getting improvement over identical model pre-trained on ImageNet of 0.0253 Top-1 score.

The difficulty of solved problem was found out by 337 Kaggle contestants that were unable to surpass the baseline of Top-1 score 0.51759 5.2.3 set for the competition. The model presented by this work have achieved late submission Top-1 score of 0.61061. At the closure of contest, the best achieved result was score of 0.53691, placing the author on 284th place ahead of 343 other competitors. The approach taken to solve this problem was compared with the approach chosen by other competitors in chapter 8. The gap of best presented solution and the winner of competition is significant. The winner scored 0.79567. However, the top ranking competitors used much more powerful hardware, training on multiple GPU's in parallel, or even using several multi-gpu setups [73].

Bibliography

[1] Sven Abels and Axel Hahn. "Reclassification of Electronic Product Catalogs: The" Apricot" Approach and Its Evaluation Results." In: *Informing Science* 9 (2006).

[2] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[3] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[4] *Kaggle - Cdiscount's Image Classification Challenge*. 2017. URL: https://www.kaggle.com/c/cdiscount-image-classification-challenge.

[5] Brian Kenji Iwana et al. "Judging a Book by its Cover". In: *arXiv preprint arXiv:1610.09204* (2016).

[6] uchidalab. *Book Dataset*. https://github.com/uchidalab/book-dataset. 2018.

[7] Cai-Nicolas Ziegler, Georg Lausen, and Lars Schmidt-Thieme. "Taxonomy-driven computation of product recommendations". In: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM. 2004, pp. 406–415.

[8] Dan Shen et al. "A study of smoothing algorithms for item categorization on e-commerce sites". In: *Neurocomputing* 92 (2012), pp. 54–60.

[9] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. "Large-scale item categorization for e-commerce". In: *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM. 2012, pp. 595–604.

[10] Hsiang-Fu Yu et al. *Product title classification versus text classification*. 2012.

[11] Jianfu Chen and David Warren. "Cost-sensitive learning for large-scale hierarchical classification". In: *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM. 2013, pp. 1351–1360.

[12] Jung-Woo Ha, Hyuna Pyo, and Jeonghee Kim. "Large-scale item categorization in e-commerce using multiple recurrent neural networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 107–115.

[13] Eli Cortez et al. "Lightweight methods for large-scale product categorization". In: *Journal of the Association for Information Science and Technology* 62.9 (2011), pp. 1839–1848.

[14] Zornitsa Kozareva. "Everyone Likes Shopping! Multi-class Product Categorization for e-Commerce." In: *HLT-NAACL*. 2015, pp. 1329–1333.

[15] Anitha Kannan et al. "Improving product classification using images". In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE. 2011, pp. 310–319.

[16] Tom Zahavy et al. "Is a picture worth a thousand words? A Deep Multi-Modal Fusion Architecture for Product Classification in e-commerce". In: *arXiv preprint arXiv:1611.09534* (2016).

[17] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[18] SA Oyewole et al. "Classification of Product Images in Different Color Models with Customized Kernel for Support Vector Machine". In: *Artificial Intelligence, Modelling and Simulation (AIMS), 2015 3rd International Conference on*. IEEE. 2015, pp. 153–158.

[19] Xing Xie et al. "Mobile search with multimodal queries". In: *Proceedings of the IEEE* 96.4 (2008), pp. 589–601.

[20] SA Oyewole and OO Olugbara. "Product image classification using Eigen Colour feature with ensemble machine learning". In: *Egyptian Informatics Journal* (2017).

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[22] Takuya Yashima et al. "Learning to Describe E-Commerce Images from Noisy Online Data". In: *Asian Conference on Computer Vision*. Springer. 2016, pp. 85–100.

[23] Stephen Zakrewsky, Kamelia Aryafar, and Ali Shokoufandeh. "Item Popularity Prediction in E-commerce Using Image Quality Feature Vectors". In: *arXiv preprint arXiv:1605.03663* (2016).

[24] Corey Lynch, Kamelia Aryafar, and Josh Attenberg. "Images don't lie: Transferring deep visual semantic features to large-scale multimodal learning to rank". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 541–548.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[26] Yann N Dauphin et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *Advances in neural information processing systems*. 2014, pp. 2933–2941.

[27] Richard S Sutton. "Two problems with backpropagation and other steepest-descent learning procedures for networks". In: *Proc. 8th annual conf. cognitive science society*. Erlbaum. 1986, pp. 823–831.

[28] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural networks* 12.1 (1999), pp. 145–151.

[29] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[30] Matthew D Zeiler. "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[31] Francois Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[32] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[33] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[34] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[35] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[36] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[37] Alex Krizhevsky and Geoffrey E Hinton. "Using very deep autoencoders for content-based image retrieval." In: *ESANN*. 2011.

[38] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[39] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[40] Kaiming He et al. "Identity mappings in deep residual networks". In: *European Conference on Computer Vision*. Springer. 2016, pp. 630–645.

[41] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[42] Saining Xie et al. "Aggregated residual transformations for deep neural networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 5987–5995.

[43] Gao Huang et al. "Deep networks with stochastic depth". In: *European Conference on Computer Vision*. Springer. 2016, pp. 646–661.

[44] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. "Fractalnet: Ultra-deep neural networks without residuals". In: *arXiv preprint arXiv:1605.07648* (2016).

[45] Gao Huang et al. "Densely connected convolutional networks". In: *arXiv preprint arXiv:1608.06993* (2016).

[46] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[47] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2818–2826.

[48] C Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv, 2016”. In: *arXiv preprint arXiv:1602.07261* ().

[49] Shaoqing Ren et al. “Faster R-CNN: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[50] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[51] Inc. MongoDB. *MongoDB for GIANT Ideas | MongoDB*. 2018. URL: https://www.mongodb.com/ (visited on 01/01/2018).

[52] pymongo. *pymongo 3.6.0*. 2018. URL: https://pypi.python.org/pypi/pymongo (visited on 01/01/2018).

[53] vfdev. *Kaggle cdiscount classifiaction challenge - random item access*. 2018. URL: https://www.kaggle.com/vfdev5/random-item-access/code (visited on 01/01/2018).

[54] Python Software Foundation. *Python Language Reference, version 3.6*. 2018. URL: https://www.python.org/ (visited on 01/01/2018).

[55] inversion. *Kaggle - Cdiscount's Image Classification Challenge - Welcome!* 2017. URL: https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/39463%5C#225384.

[56] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *arXiv preprint arXiv:1612.08242* (2016).

[57] Wei-Ta Chu and Hung-Jui Guo. “Movie Genre Classification based on Poster Images with Deep Neural Networks”. In: (2017).

[58] Kaiming He et al. “Mask r-cnn”. In: *arXiv preprint arXiv:1703.06870* (2017).

[59] raghakot. *keras-resnet*. https://github.com/raghakot/keras-resnet. 2018.

[60] Inc. Anaconda. *Conda - conda documentation*. 2018. URL: https://conda.io/docs/ (visited on 01/01/2018).

[61] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed <today>]. 2001–. URL: http://www.scipy.org/.

[62] matterport. *Mask R-CNN for Object Detection and Segmentation*. https://github.com/matterport/Mask_RCNN. 2018.

[63] *MATLAB Optimization Toolbox*. The MathWorks, Natick, MA, USA. 2016.

[64] Hang Su et al. “Multi-view convolutional neural networks for 3d shape recognition”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 945–953.

[65] Lam Dang. *Kaggle - Cdiscount's Image Classification Challenge - Our 7th place solution*. 2017. URL: https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/45737.

[66] Antoine Miech, Ivan Laptev, and Josef Sivic. “Learnable pooling with Context Gating for video classification”. In: *arXiv preprint arXiv:1706.06905* (2017).

[67] Relja Arandjelovic et al. "NetVLAD: CNN architecture for weakly supervised place recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5297–5307.

[68] Antoine Miech, Ivan Laptev, and Josef Sivic. "Learnable pooling with Context Gating for video classification". In: *arXiv:1706.06905* (2017).

[69] Radu Stoicescu. *Kaggle - Cdiscount's Image Classification Challenge - Single best model?* 2017. URL: https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/45729.

[70] NighTurs. *kaggle-cdiscount-image-classification*. https://github.com/NighTurs/kaggle-cdiscount-image-classification. 2017.

[71] Heng CherKeng. *Kaggle - Cdiscount's Image Classification Challenge - any one trying multiple classifier approach?* 2017. URL: https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/41632.

[72] CdiscountDS. *Kaggle - Cdiscount's Image Classification Challenge - Can We use dataset for Academic research purpose ?* 2017. URL: https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/40393#226910.

[73] n01z3. *Kaggle - Cdiscount's Image Classification Challenge - My solution [9th place private]*. 2017. URL: https://www.kaggle.com/c/cdiscount-image-classification-challenge/discussion/45721.